-74

55087
~~53000~~

Supreme Court, U.S.
FILED
NOV 5 - 2007
OFFICE OF THE CLERK

79pg (15

No. 06-937

In the
Supreme Court of the United States

QUANTA COMPUTER, INC., QUANTA COMPUTER USA, INC., Q-LITY COMPUTER, INC.,
PETITIONERS,

v.

LG ELECTRONICS, INC.,
RESPONDENT.

ON WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

SUPPLEMENT TO THE JOINT APPENDIX
(Pages 199 to 267)

Counsel for Petitioner

MAUREEN E. MAHONEY
Counsel of Record
J. SCOTT BALLENGER
MELISSA B. ARBUS
ANNE W. ROBINSON
LATHAM & WATKINS LLP
555 11TH STREET, N.W.
SUITE 1000
WASHINGTON, DC 20004
(202) 637-2200
(202) 637-2201 (FAX)
maureen.mahoney@lw.com

Counsel for Respondent

CARTER G. PHILLIPS
Counsel of Record
VIRGINIA A. SEITZ
JEFFREY T. GREEN
JEFFREY P. KUSHAN
SIDLEY AUSTIN LLP
1501 K STREET, NW
WASHINGTON, DC 20005
(202) 736-8015
(202) 736-8711 (FAX)
cphillips@sidley.com

Additional Counsel Listed on Inside Cover

Additional Counsel for Petitioner

TERRENCE D. GARNETT
VINCENT K. YIP
PETER WIED
PAUL, HASTINGS, JANOFSKY & WALKER LLP
515 SOUTH FLOWER STREET
25TH FLOOR
LOS ANGELES, CA 90071
(213) 683-6000

MAXWELL A. FOX
PAUL, HASTINGS, JANOFSKY & WALKER LLP
34 F ARK MORI BUILDING
P.O. BOX 577
12-32 AKASAKA 1-CHOME
MINATO-KU, TOKYO 107-6034
JAPAN
813 6229 6100

JOINT APPENDIX
TABLE OF CONTENTS – VOL. I

Page

United States Court of Appeals for the Federal Circuit Relevant Docket Entries: *LG Electronics v. Bizcom Electronics*, No. 05-1261 1

United States District Court for the Northern District of California Relevant Docket Entries: *LG Electronics, Inc. v. Q-Lity Computer Inc., et al.*, No. 4:01-cv-02187-CW 7

Complaint, *LG Electronics Inc. v. Q-Lity Computer Inc., Quanta Computer, Inc. and Quanta Computer USA, Inc.* (N.D. Cal. Apr. 16, 2001) 134

September 4, 2000 Letter from John Matheson to David Shing, attached to Declaration of David M. Morris in Support of LGE's Cross-Motion for Partial Summary Judgment (N.D. Cal. May 3, 2002) (A 3960[1]) (bates number omitted in printing) 143

JOINT APPENDIX
TABLE OF CONTENTS – VOL. II
(UNDER SEAL)

Page

Patent Cross License Agreement Between LG Electronics, Inc. and Intel Corporation, Exhibit A to Declaration of Maxwell A. Fox in Support of Defendants Q-Lity Computer Inc., Quanta Computer Inc., and Quanta Computer USA, Inc.'s Motion For Partial Summary Judgment (N.D. Cal. May 3, 2002) (A 3605-3619) (bates numbers omitted in printing) 145

Master Agreement with Attachments A-C, Exhibit B to Declaration of Maxwell A. Fox in Support of Defendants Q-Lity Computer Inc., Quanta Computer Inc., and Quanta Computer USA, Inc.'s Motion For Partial Summary Judgment (N.D. Cal. May 3, 2002) (A 3621-3632) (bates numbers omitted in printing) 174

---

[1] "A ___" refers to the Joint Appendix filed in the Federal Circuit.

SUPPLEMENT TO THE JOINT APPENDIX
TABLE OF CONTENTS

Page

United States Patent No. 4,939,641, Multi-Processor System with Cache Memories (July 3, 1990) (A 139 to A 159) 199

United States Patent No. 5,077,733, Priority Apparatus Having Programmable Node Dwell Time (Dec. 31, 1991) (A 160 to A 183) 220

United States Patent No. 5,379,379, Memory Control Unit with Selective Execution of Queued Read and Write Requests (Jan. 3, 1995) (A 204 to A 227) 244

ITEMS PREVIOUSLY REPRODUCED

In accordance with Supreme Court Rule 26.1, the following items have been omitted in printing this joint appendix because they appear on the following pages of the appendix to the Petition for a Writ of Certiorari (Nov. 30, 2006)

Page

Opinion, *LG Electronics, Inc. v. Bizcom Electronics, Inc., et al.*, Nos. 05-1261, 05-1262, 05-1263, 05-1264, 05-1302, 05-1303, 05-1304 (Fed. Cir. July 7, 2006) 1a

Order on Cross-Motions for Summary Judgment, *LG Electronics, Inc. v. Asustek Computer, Inc., et al.*, Nos. C 01-00326 CW, C 01-1375 CW, C 01-01594 CW, C 01-02187 CW, C 01-01552 CW (N.D. Cal. Aug. 20, 2002) 26a

Order Granting in Part Defendants' Motion for Summary Adjudication and Granting in Part Plaintiff's Motion for Summary Adjudication, *LG Electronics, Inc. v. Asustek Computer, Inc., et al.*, Nos. C 01-00326 CW, C 01-1375 CW, C 01-01594 CW, C 01-02187 CW, C 01-01552 CW (N.D. Cal. Feb. 6, 2003) 52a

ITEMS PREVIOUSLY REPRODUCED—Continued

Page

Order Addressing Cross Motions for Summary Judgment and Granting Defendants' Motion to Amend, *LG Electronics, Inc. v. Bizcom Electronics, Inc., et al.*, Nos. C 01-00326 CW, C 01-1375 CW, C 01-01594 CW, C 01-02187 CW, C 01-01552 CW (N.D. Cal. Nov. 30, 2004) 62a

Order Denying Rehearing, *LG Electronics, Inc. v. Bizcom Electronics, Inc., et al.*, Nos. 05-1261, 05-1262, 05-1263, 05-1264, 05-1302, 05-1303, 05-1304 (Fed. Cir. Sept. 1, 2006) 82a

United States Patent [19]
Schwartz et al.

[11] **Patent Number: 4,939,641**
[45] **Date of Patent: Jul. 3, 1990**

[54] **MULTI-PROCESSOR SYSTEM WITH CACHE MEMORIES**

[75] Inventors: **Martin J. Schwartz, Worcester; Robert D. Becker, Shirley, both of Mass.**

[73] Assignee: **Wang Laboratories, Inc., Lowell, Mass.**

[21] Appl. No.: **213,356**

[22] Filed: **Jun. 30, 1988**

[51] Int. Cl.[5] G06F 13/00
[52] U.S. Cl. 364/200; 364/243.4; 364/243.41; 364/259.2
[58] Field of Search ... 364/200 MS File, 900 MS File

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,735,360	5/1973	Anderson et al.	364/200
3,761,883	9/1973	Alvarez et al.	364/200
3,845,474	10/1974	Lange et al.	364/200
3,967,247	6/1976	Anderson et al.	364/200
4,167,782	9/1979	Joyce et al.	364/200
4,394,732	7/1983	Swenson	364/200
4,442,487	4/1984	Fletcher et al.	364/200
4,464,712	8/1984	Fletcher	364/200
4,527,238	7/1985	Ryan et al.	364/200
4,531,799	11/1985	Ryan et al.	364/200
4,602,368	6/1986	Circello et al.	364/200
4,675,811	6/1987	Kishi et al.	364/200
4,685,082	8/1987	Chang et al.	365/49
4,811,209	3/1989	Rabuzzin	364/200

OTHER PUBLICATIONS

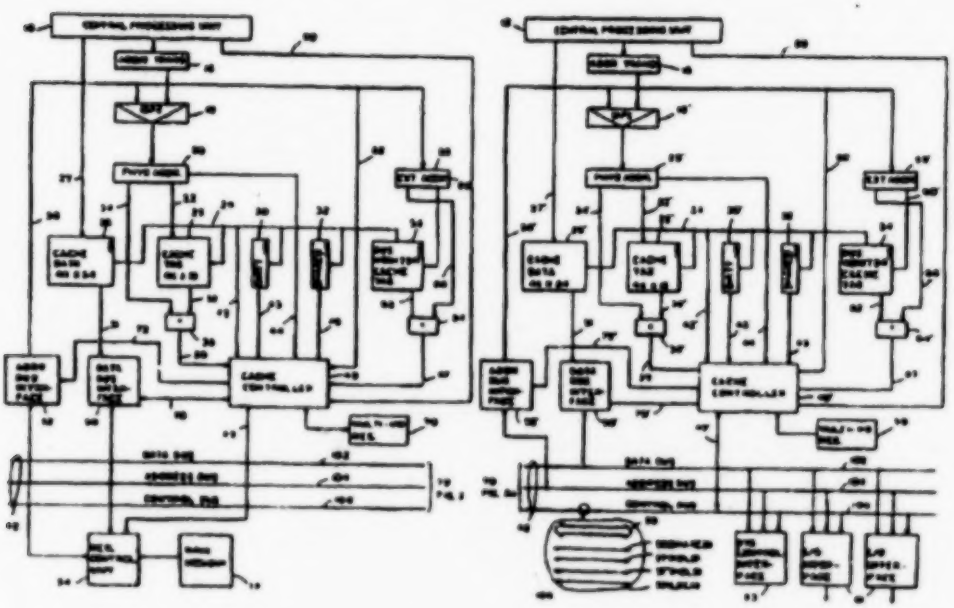
Cousier & Feautrier, A New Solution to Coherence Problems in Multicache Systems, IEEE Transactions on Computers, vol. C-27, No. 12, 9/78, pp. 1112-1118.
"Cache Memories", by Alan Jay Smith, Computer Surveys, vol. 14, No. 3, Sep. 1982.
"16-KBobyte Cache/Memory Management System", Motorola SemiConductor Technical Data Brochure.

Primary Examiner—David Y. Eng
Attorney, Agent, or Firm—Michael H. Shanahan

[57] **ABSTRACT**

A system is described wherein a CPU, a main memory means and a bus means are provided. Cache memory means is employed to couple the CPU to the bus means and is further provided with means to indicate the status of a data unit stored within the cache memory means. One status indication tells whether the contents of a storage position have been modified since those contents were received from main memory and another indicates whether the contents of the storage position may be present elsewhere memory means. Control means are provided to assure that when a data unit from a CPU is received and stored in the CPU's associated cache memory means, which data unit is indicated as being also stored in a cache memory means associated with another CPU, such CPU data unit is also written into main memory means. During that process, other cache memory means monitor the bus means and update its corresponding data unit. Bus monitor means are provided and monitor all writes to main memory and reads from main memory to aid in the assurance of system-wide data integrity.

16 Claims, 10 Drawing Sheets



This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™ a product of Patent Logistics, LLC Page 1 of 21

BEST AVAILABLE COPY

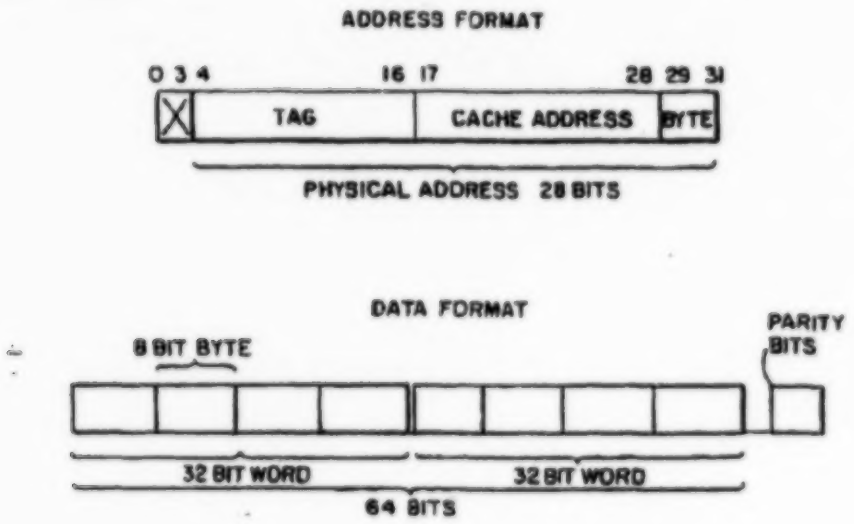


FIG. 1

This PDF of U.S. Utility Patent 4939641 provided by PatentFetcher™, a product of Patent Logistics, LLC - Page 2 of 21

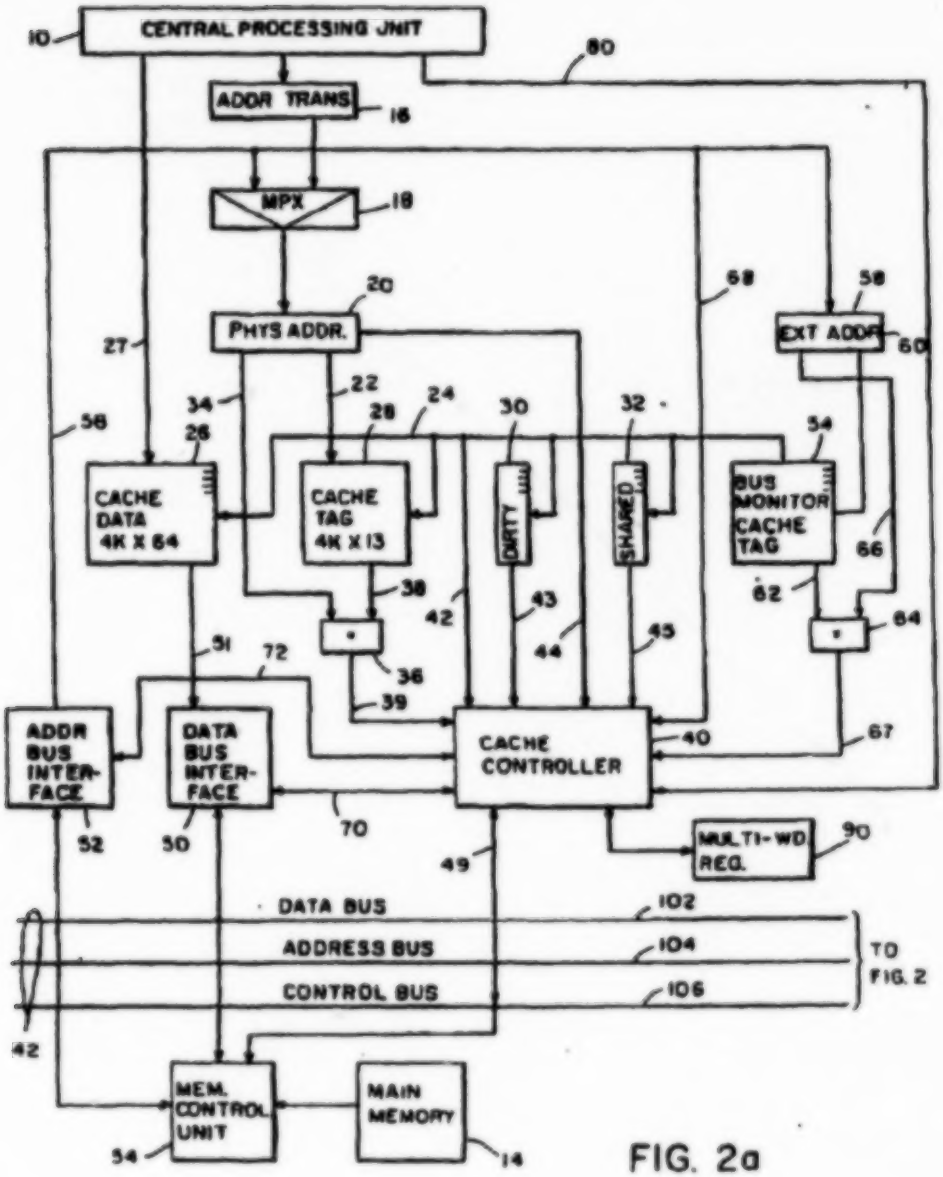


FIG. 2a

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 3 of 21

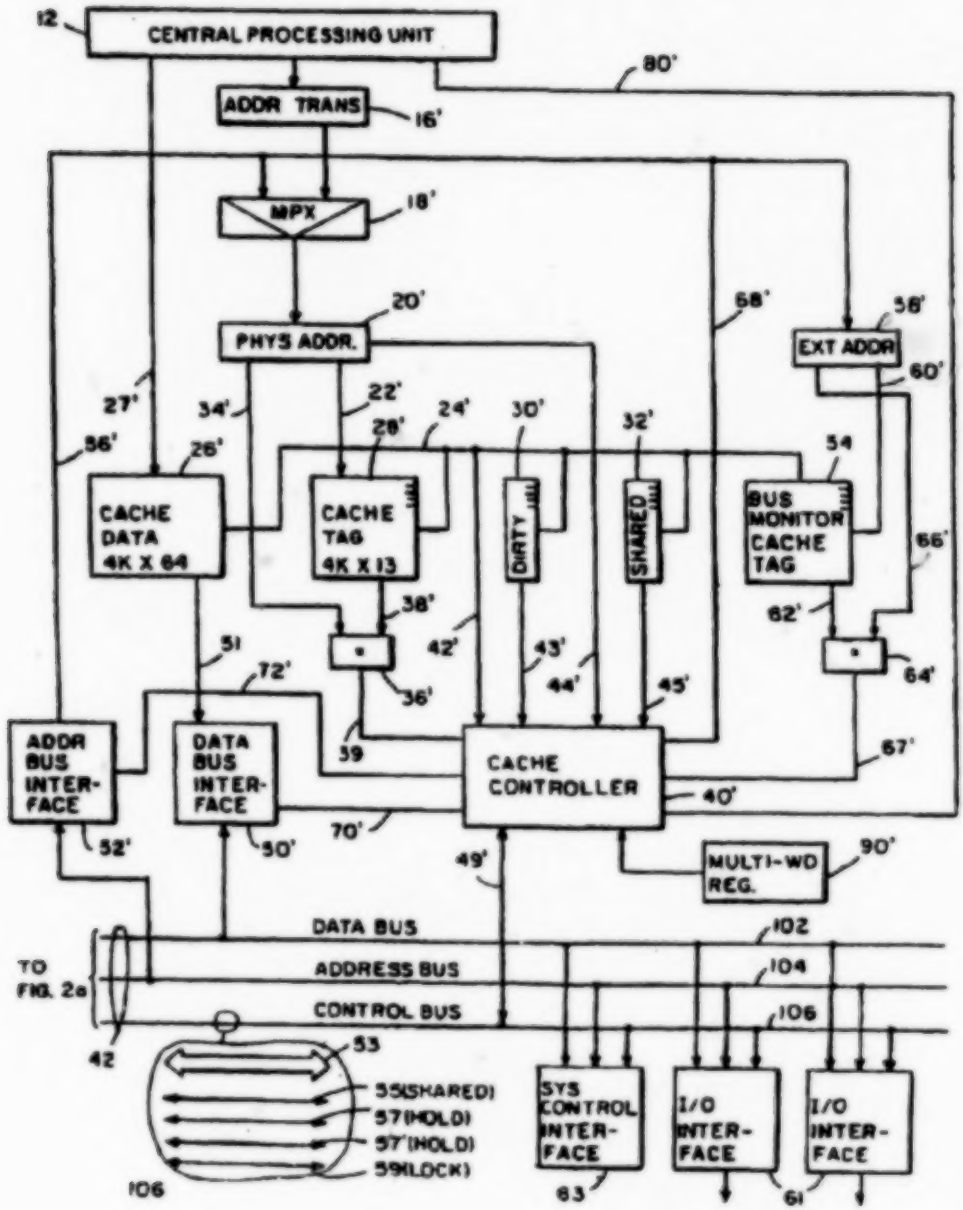


FIG. 2b

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 4 of 21

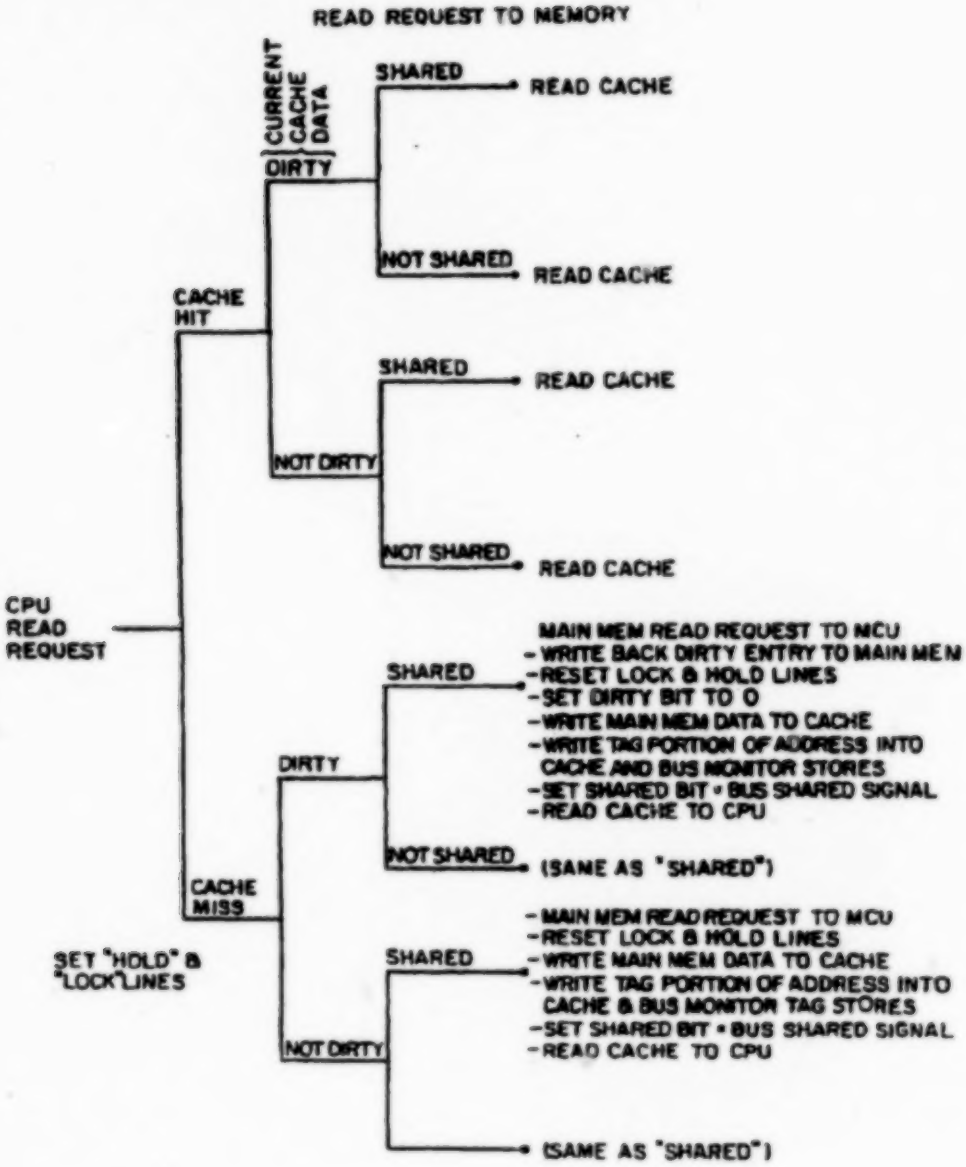

READ REQUEST TO MEMORY
CURRENT CACHE DATA
DIRTY
SHARED
READ CACHE
NOT SHARED
READ CACHE
CACHE HIT
SHARED
READ CACHE
NOT DIRTY
NOT SHARED
READ CACHE
CPU READ REQUEST
MAIN MEM READ REQUEST TO MCU
- WRITE BACK DIRTY ENTRY TO MAIN MEM
- RESET LOCK & HOLD LINES
- SET DIRTY BIT TO 0
- WRITE MAIN MEM DATA TO CACHE
- WRITE TAG PORTION OF ADDRESS INTO CACHE AND BUS MONITOR STORES
- SET SHARED BIT = BUS SHARED SIGNAL
- READ CACHE TO CPU
SHARED
DIRTY
NOT SHARED
(SAME AS "SHARED")
CACHE MISS
SET "HOLD" & "LOCK" LINES
- MAIN MEM READ REQUEST TO MCU
- RESET LOCK & HOLD LINES
- WRITE MAIN MEM DATA TO CACHE
- WRITE TAG PORTION OF ADDRESS INTO CACHE & BUS MONITOR TAG STORES
- SET SHARED BIT = BUS SHARED SIGNAL
- READ CACHE TO CPU
SHARED
NOT DIRTY
(SAME AS "SHARED")

FIG. 3

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 5 of 21

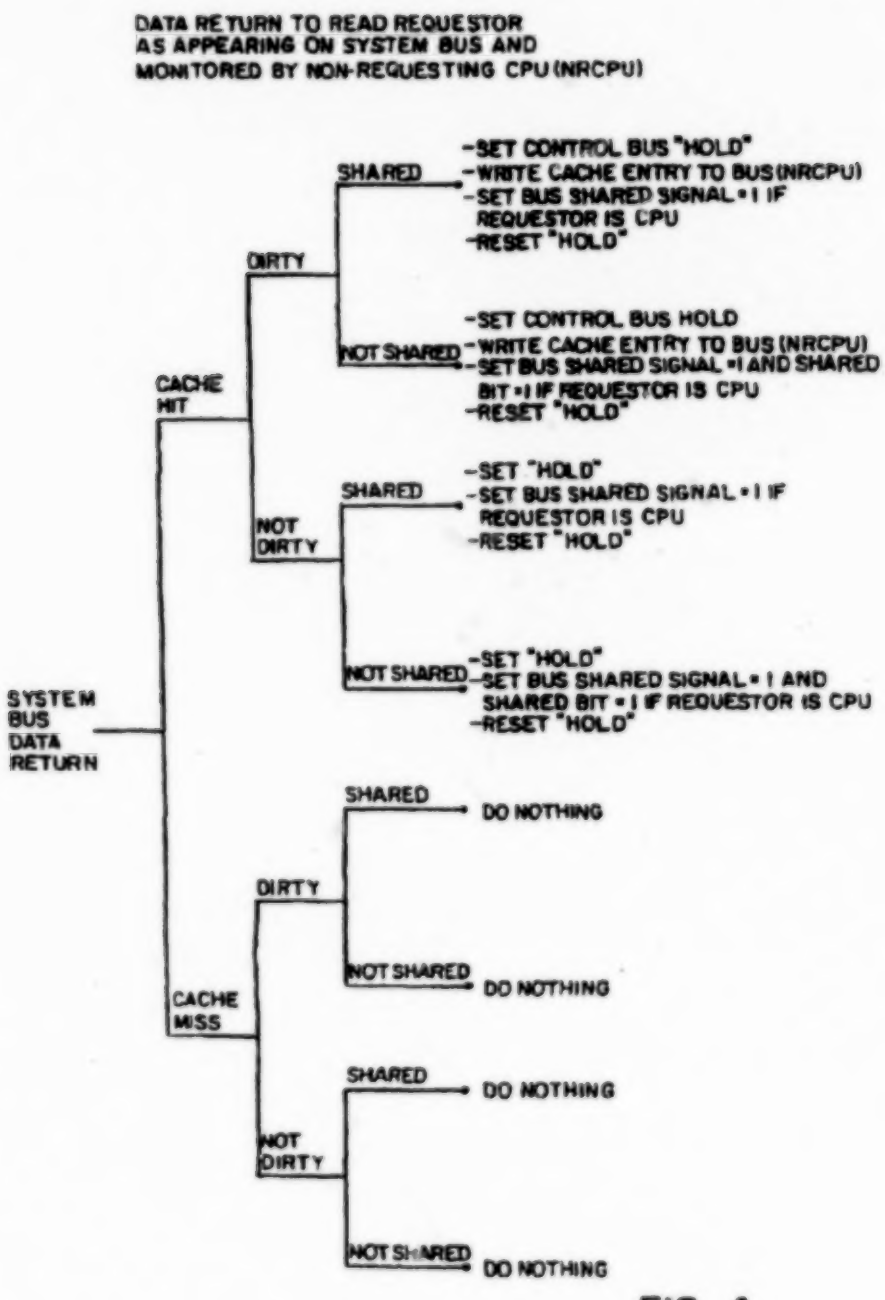

DATA RETURN TO READ REQUESTOR
AS APPEARING ON SYSTEM BUS AND
MONITORED BY NON-REQUESTING CPU (NRCPU)
SYSTEM BUS DATA RETURN
CACHE HIT
CACHE MISS
DIRTY
NOT DIRTY
SHARED
NOT SHARED
-SET CONTROL BUS "HOLD"
-WRITE CACHE ENTRY TO BUS (NRCPU)
-SET BUS SHARED SIGNAL = 1 IF REQUESTOR IS CPU
-RESET "HOLD"
-SET CONTROL BUS HOLD
-WRITE CACHE ENTRY TO BUS (NRCPU)
-SET BUS SHARED SIGNAL = 1 AND SHARED BIT = 1 IF REQUESTOR IS CPU
-RESET "HOLD"
-SET "HOLD"
-SET BUS SHARED SIGNAL = 1 IF REQUESTOR IS CPU
-RESET "HOLD"
-SET "HOLD"
-SET BUS SHARED SIGNAL = 1 AND SHARED BIT = 1 IF REQUESTOR IS CPU
-RESET "HOLD"
DO NOTHING
DO NOTHING
DO NOTHING
DO NOTHING

FIG. 4

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 6 of 21

CPU WRITE TO MEMORY
CPU WRITE
CACHE HIT
CACHE MISS
SET "LOCK" & "HOLD" LINES
CURRENT CACHE DATA
DIRTY
NOT DIRTY
SHARED
NOT SHARED
UPDATE CACHE
WRITE THROUGH CACHE ENTRY TO MAIN MEM
SET DIRTY BIT = 0
UPDATE CACHE
(DIRTY BIT REMAINS = 1)
UPDATE CACHE
WRITE THROUGH CACHE ENTRY TO MAIN MEM
(DIRTY BIT REMAINS = 0)
-UPDATE CACHE
-SET DIRTY BIT = 0
-MAIN MEM READ REQUEST TO MCU
-WRITE BACK DIRTY ENTRY TO MAIN MEMORY
-SET DIRTY BIT = 0
-RESET LOCK & HOLD
-WRITE MAIN MEM DATA TO CACHE
-WRITE TAG PORTION OF ADDRESS INTO CACHE AND BUS MONITOR TAG STORES
-UPDATE CACHE BY WRITING CPU DATA OVER MAIN MEM DATA. SET DIRTY BIT = 1
-SET SHARED BIT = BUS SHARED SIGNAL
-IF SHARED BIT = 1, WRITE THROUGH CPU ENTRY TO MAIN MEM & RESET DIRTY BIT = 0
-IF SHARED BIT = 0, DIRTY BIT REMAINS = 1)
(SAME AS "SHARED")
-MAIN MEM READ REQUEST TO MCU
-RESET LOCK & HOLD
-WRITE MAIN MEM DATA TO CACHE
-UPDATE CACHE BY WRITING CPU DATA OVER MAIN MEM DATA; SET DIRTY BIT = 1
-SET SHARED BIT = BUS SHARED SIGNAL
-IF SHARED BIT = 1, WRITE THROUGH CPU ENTRY TO MAIN MEM; RESET DIRTY BIT = 0
-IF SHARED BIT = 0, DIRTY BIT REMAINS = 1
(SAME AS "SHARED")

FIG. 5

This PDF of U.S. Utility Patent 4939641 provided by PatentFetcher™, a product of Patent Logistics, LLC - Page 7 of 21

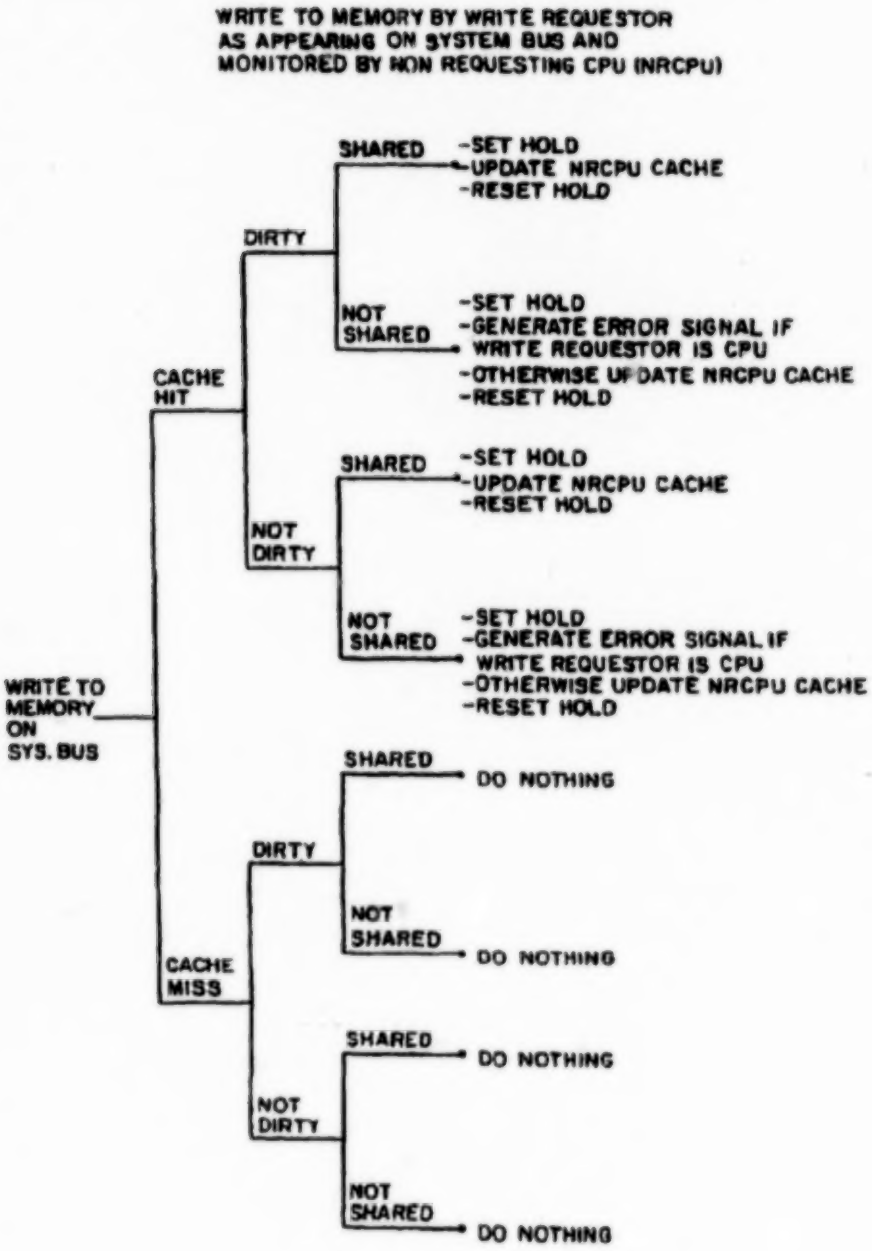


FIG. 6

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 8 of 21

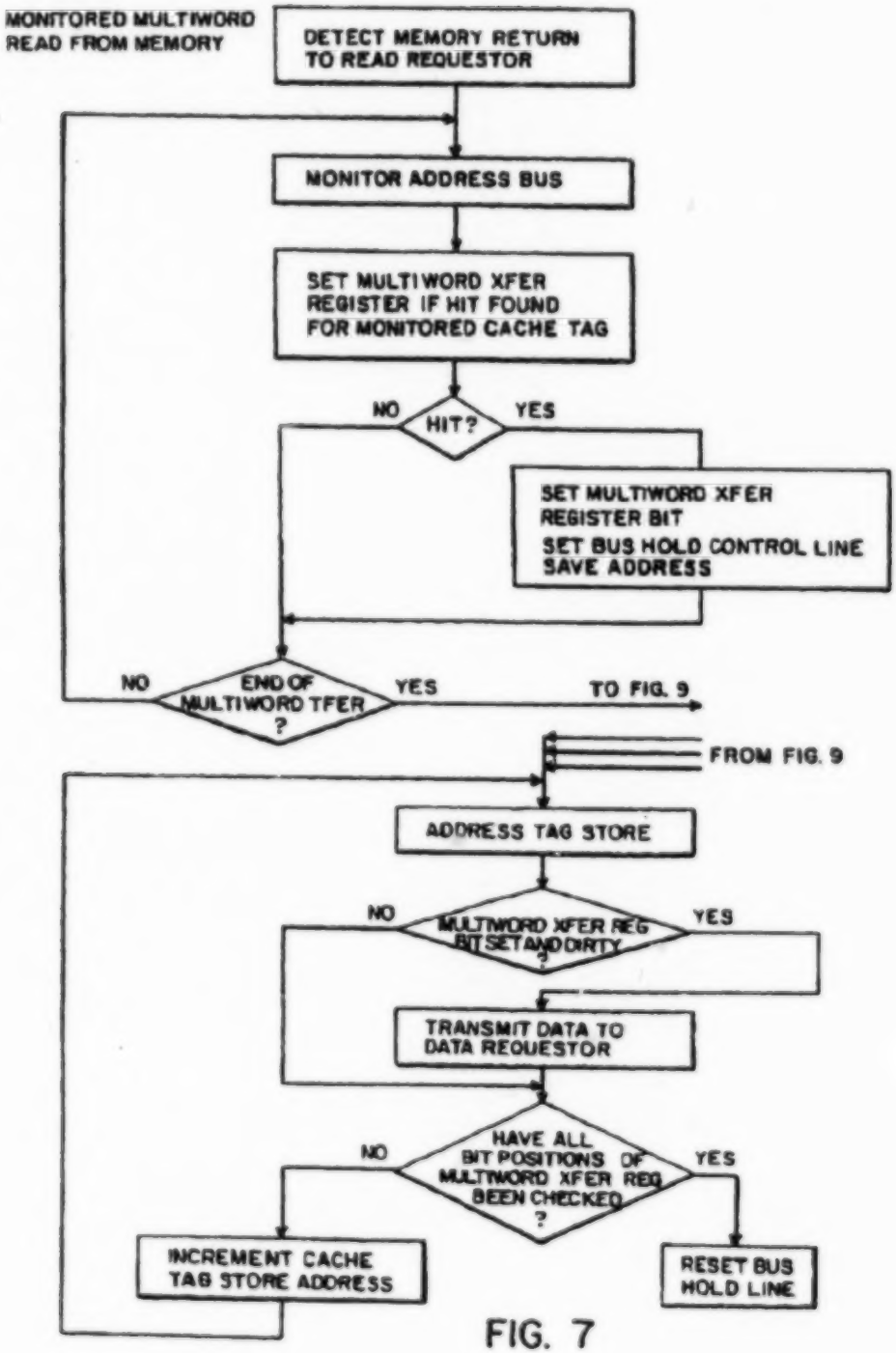


FIG. 7

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 9 of 21

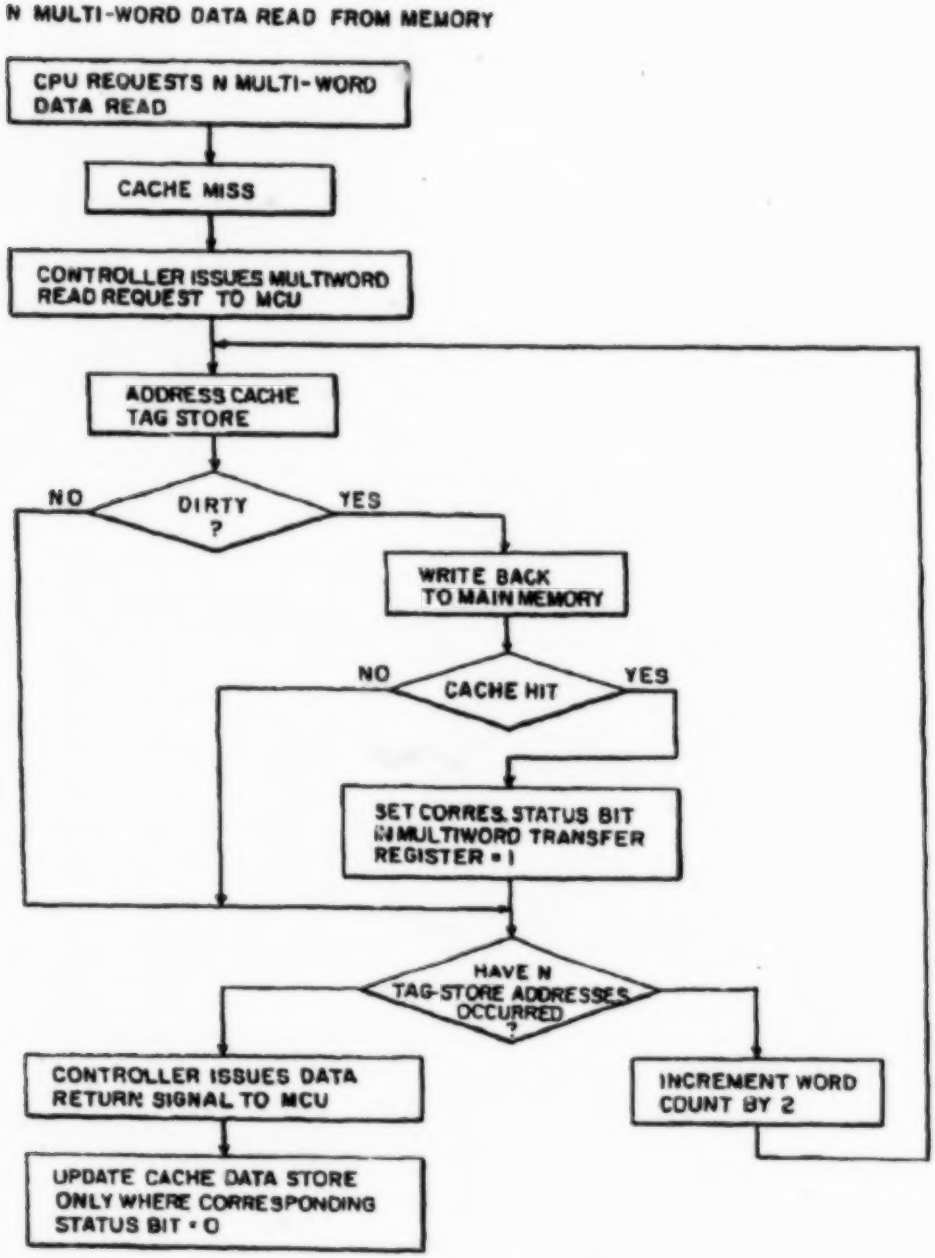

N MULTI-WORD DATA READ FROM MEMORY
CPU REQUESTS N MULTI-WORD DATA READ
CACHE MISS
CONTROLLER ISSUES MULTIWORD READ REQUEST TO MCU
ADDRESS CACHE TAG STORE
NO
DIRTY ?
YES
WRITE BACK TO MAIN MEMORY
NO
CACHE HIT
YES
SET CORRES. STATUS BIT IN MULTIWORD TRANSFER REGISTER = 1
HAVE N TAG-STORE ADDRESSES OCCURRED ?
CONTROLLER ISSUES DATA RETURN SIGNAL TO MCU
INCREMENT WORD COUNT BY 2
UPDATE CACHE DATA STORE ONLY WHERE CORRESPONDING STATUS BIT = 0

FIG. 8

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™ a product of Patent Logistics, LLC Page 10 of 21

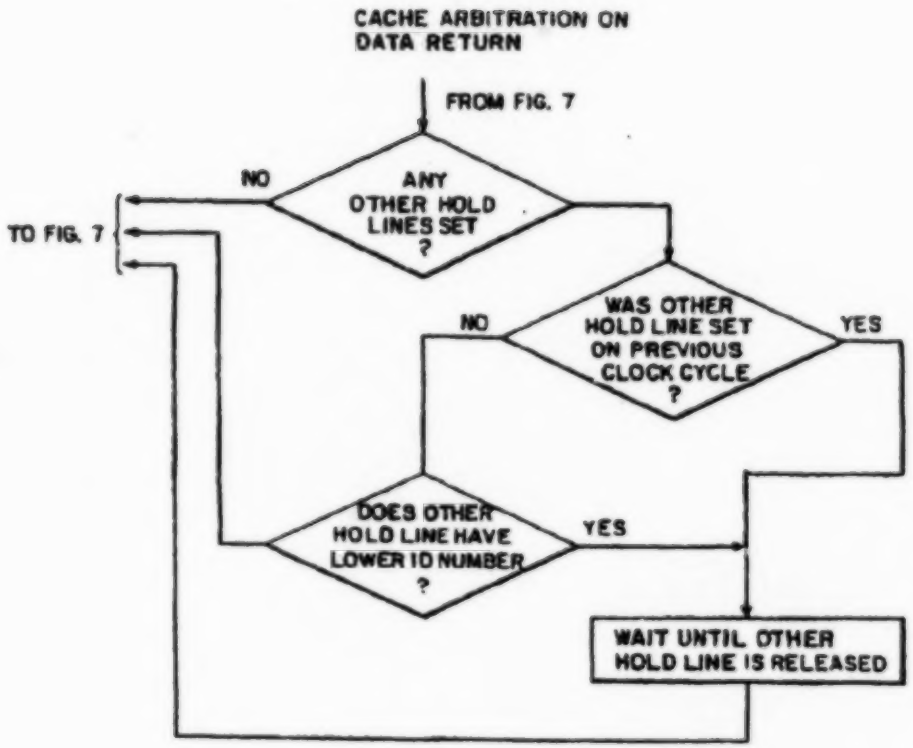

CACHE ARBITRATION ON DATA RETURN
FROM FIG. 7
ANY OTHER HOLD LINES SET ?
NO
TO FIG. 7
WAS OTHER HOLD LINE SET ON PREVIOUS CLOCK CYCLE ?
NO
YES
DOES OTHER HOLD LINE HAVE LOWER ID NUMBER ?
YES
WAIT UNTIL OTHER HOLD LINE IS RELEASED

FIG. 9

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 11 of 21

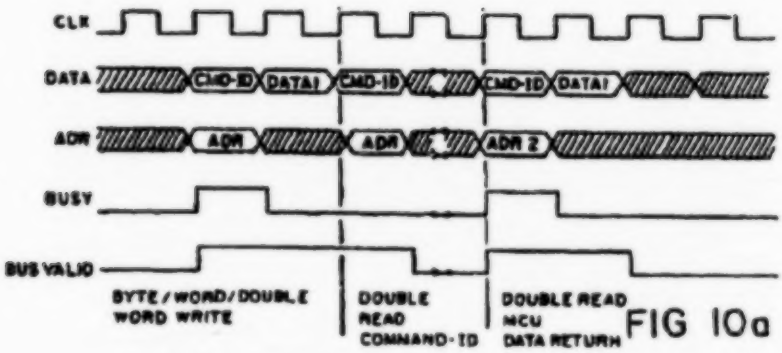


FIG 10a

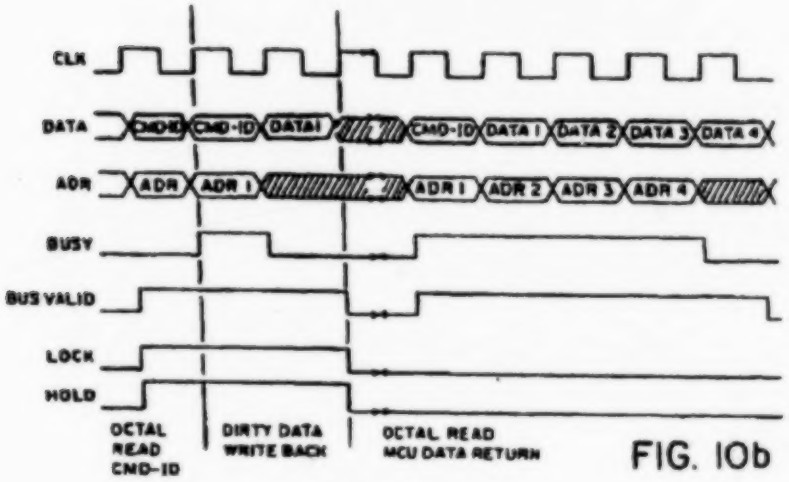


FIG. 10b

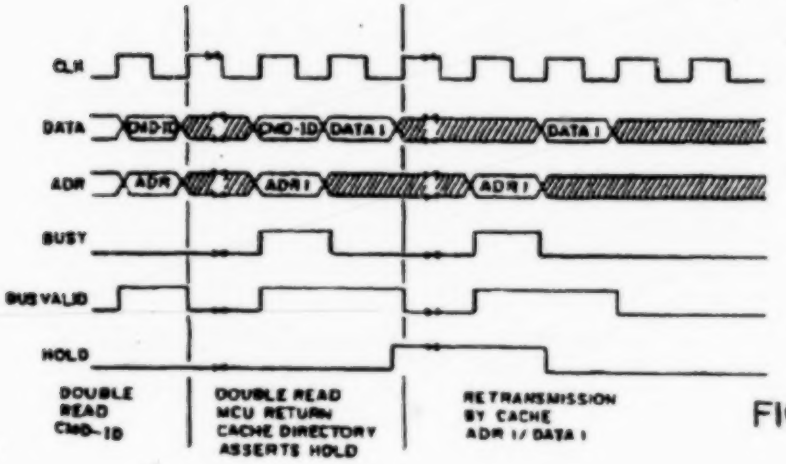


FIG. 10c

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 12 of 21

MULTI-PROCESSOR SYSTEM WITH CACHE MEMORIES

I. FIELD OF THE INVENTION

This invention relates to multi-processor computing systems which employ cache memories and more particularly to means and methods for maintaining data integrity therein.

BACKGROUND OF THE INVENTION

The use of high speed, small capacity cache memory systems is well known. While there are a variety of cache memory systems, several of the better known are termed "write through" caches and "write back" caches. In a write through cache, data is written into main memory at the same time it is written into the cache. Thus, the cache always contains identical data to that stored in main memory and data integrity (coherency) is maintained throughout the system. The problem with write through architecture is that it creates an excessive amount of bus traffic, in that a write to main memory occurs every time there is a write to the write through cache. A positive aspect of this architecture is that it is always known where the most updated data resides, i.e., everywhere.

In an architecture employing a write back cache, the amount of traffic on the bus is significantly reduced. Initially, data is written into the write back cache from main memory and is then used by the central processing unit (CPU) for operations. When the CPU writes the data back into the cache and assuming it has been modified, a "dirty" bit is set to indicate that the data is now unique to that cache and is different from that which resides in main memory. In such a cache, in general, no immediate effort is made to write the revised data into the main memory to maintain data integrity. Obviously then, it is the dirty bit which is critical to the maintenance of data coherence. So long as a write back cache is utilized with only one processor, data management is straight forward. However, when more than one central processor uses the same main memory, data management problems multiply.

In such systems, there is often more than one cache memory present. Each entry position in a cache is provided with a valid/invalid bit. If a CPU sees that another cache is writing to memory and finds that its cache contains an identical data address, it invalidates its own cache entry rather than updating it. Thus, when a CPU accesses its cache at that particular data address, it finds an invalid entry and is redirected to main memory, a time consuming process.

It is therefore an object of this invention to maintain data integrity in a multi-processor/cache environment without requiring exclusive access to main memory.

It is another object of this invention to enable a number of CPU's to access data via their associated cache memories with each processor knowing that it is always accessing the most updated data.

It is still another object of this invention to provide a multi-processor system employing write back caches wherein main memory accesses are minimized while simultaneously maintaining data integrity throughout the system.

It is a further object of this invention to provide a cache memory system wherein the use of valid/invalid data indicators are avoided.

SUMMARY OF THE INVENTION

A multi-processing system is described wherein at least two CPU's, a main memory means and a bus means are provided. Cache memory means are employed to couple each CPU to the bus means and are further provided with means to indicate the status of a data unit stored within the cache memory means. One status indication tells whether the contents of a storage position have been modified since those contents were received from main memory and another indicates whether the contents of the storage position may be present in another cache memory means. Control means are provided to assure that when a data unit from a CPU is received and stored in the CPU's associated cache memory means, which data unit is indicated as being also stored in a cache memory means associated with another CPU, such CPU data unit is also written into main memory means. During that process, other cache memory means monitors the bus means and updates its corresponding data unit. Bus monitor means are provided for each cache memory means and monitor all writes to main memory and reads from main memory to aid in the assurance of system-wide data integrity.

DESCRIPTION OF THE DRAWINGS

FIG. 1 shows both the address and data formats for the data processing system embodying the invention.

FIGS. 2a and 2A in combination, illustrate a high level block diagram of the invention.

FIG. 3 is a decision tree indicating the sequence of events which occur during a CPU read request to memory.

FIG. 4 is a decision tree indicating the sequence of events initiated when a non-requesting CPU/cache monitors on the system bus a data return to a read-requestor.

FIG. 5 is a decision tree indicating the sequence of events which occur during a CPU write to memory.

FIG. 6 is a decision tree which indicates the sequence of events which occur when a non-requesting CPU/cache monitors a write to memory by a write requestor.

FIG. 7 is a flow chart indicating the sequence of events which occurs when a non-requesting CPU/cache monitors a multi-word read from memory to a requesting CPU/cache.

FIG. 8 is a flow chart illustrating a multi word data read from memory.

FIG. 9 is a flow chart illustrating the arbitration which occurs between caches on a data return.

FIGS. 10a, 10b and 10c are timing diagrams useful in understanding the invention.

DETAILED DESCRIPTION OF THE INVENTION

The data processing system of this invention employs address and data formats shown in FIG. 1. A byte comprises 8 bits and a word 4 bytes. Each position in memory is adapted to store two words ("double word") and is addressable at the byte level. Any byte may be individually addressed, read and/or written as can any combination of bytes, words and double words. Normally, unless otherwise instructed, data transfers in this system comprise double words.

With respect to address formats, 28 bits are actively used and comprise a physical address in main memory where a subset of data (e.g. a byte) is to be found. Portions of the physical address are employed to access and identify positions within a write-back cache memory.

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 13 of 21

Each cache uses 12 bits of the physical address, (i.e., bits 4–16) to identify a data word and those 12 bits are called the tag. Another portion of the physical address, i.e., bits 17–28, is employed to provide an address to a storage position within the cache memory (called the cache address). Thus, while the cache address references a storage position within the cache memory, the tag identifies data actually stored at that storage position.

An access into a cache memory commences with the arrival of a physical address from the CPU. The cache employs the cache address to access one of its memory positions. Subsequently, the tag portion of the received physical address is compared with the tag portion of the data stored at the cache address to determine if there is a match, thus indicating that a "hit" has occurred and the proper data to be either accessed, written over or otherwise altered or utilized is the data actually designated by the physical address. If no match is found, (i.e., a "miss") the typical response of the system is to generate a read request to main memory for the requested data.

As above stated, the data format, is two words in length, with each word being 32 bits and comprising 4 eight bit bytes. There are also parity bits associated therewith but these are not relevant to a discussion of this invention.

It should be kept in mind during the following description, that the invention maintains data integrity by assuring that cache data is always the most up-to-date in the system. Thus, there never is a "valid" or "invalid" indication with respect to any cache data as it is always assured that if data is provided by a cache, that it invariably is valid (i.e. most up-to-date).

By contrast, cache systems which employ valid and invalid indicators unnecessarily multiply the number of cache misses which occur when an invalid data unit is addressed. In such systems, a valid data unit is fetched, placed in the cache and then provided to the requestor. This invention avoids those unnecessary memory cycles; avoids the need for any directory or other listing of valid and invalid data and avoids any requirement that its memory controller keep records of cache operations.

In the cache memories of this invention, indications are associated with each cache data word of its state of modification (if any) since being accessed from main memory, (e.g. "dirty" or "not dirty") and whether it also may reside in another cache store (e.g. "shared" or "not shared"). These indications enable the system to continuously update the cache data words and assures that they are updated when they are called for. Even in the case where a cache store is found not to have the requested data, if such data is resident in another cache store, provisions are made for its immediate transfer to the requested cache store.

The invention further avoids the necessity for the provision of special procedures when more than one CPU wishes to simultaneously modify bytes in a single word. This invention accomplishes such modifications during the normal course of its operations.

Since, by definition, data in the caches of this system is the most up-to-date, when system operation is initiated, each of the caches must be filled with data. This is accomplished by arbitrarily selecting data from main memory and writing it into each cache store, it being remembered, that at system start-up, main memory data is considered most up-to-date. The system then commences further operations as described hereinafter.

Turning to FIG. 2, a high level block diagram is shown of a multi processor system that includes at least two CPU's 10 and 12. Each CPU typically operates with a virtual addressing system; however, when internal CPU and memory operations are carried out, the virtual addresses are converted to physical addresses which indicate the actual address positions within main memory 14 where the data designated by the address is stored. The address conversion occurs in address translators 16 and 16' with the resulting physical address being passed via multiplexors 18 and 18' to physical address registers 20 and 20'. As each of the CPU's, along with their associated cache memory systems, are identical, CPU 10 will hereinafter be described in detail with it being understood that CPU 12 is identical and is numbered in an identical manner with prime numbers.

The portion of the physical address resident in physical address register 20 which corresponds to the cache address (i.e. bits 17–28) is fed via cable 22 to cache bus 24. Also connected to cache bus 24 is cache data store 26, cache tag store 28, "dirty" bit store 30, "shared" bit store 32 and bus monitor cache tag store 34. A cable 27 provides for direct entry of data from CPU 10 to cache data store 26. Cache data store 26 is typically a 32K byte memory that is 4K lines deep by 64 bits wide (8 bytes). Each line of this cache stores a double word. Cache tag store 28 is typically 4K lines deep by 13 bits wide and stores the tag portion of the physical address, which tag defines the address where the corresponding data is stored in cache data store 26. It should be remembered that each double word of data has associated with it a physical address which is indicative of the position in main memory 14 where that double word is stored. Thus, each line of cache tag store 28 corresponds in position to a data storage address in cache data store 26 and contains the tag portion of the physical address associated with the data stored at such cache data store address.

Dirty bit register 30 and shared bit register 32 are each also 4K lines deep. A "1" bit on any line of dirty bit register 30 indicates that data in data in the corresponding line of cache data store 26 is more up to date than data in the corresponding storage location in main memory 14. A "0" setting on any line of dirty bit register 30 indicates that the corresponding entry in cache data store 26 is the same as that appearing at that location in main memory (with one exception to be hereinafter described). A bit set to 1 on any line of shared bit register 32 indicates that another cache in the system might contain the same entry. A bit set to 0 on any line of shared bit register 32 indicates that no other cache has the same entry.

Referring back to physical address register 20, cable 34 accesses the tag portion (bits 4–16) of the physical address and presents it to comparator 36. Cable 38 from cache tag store 28 also presents to comparator 36 a selected tag stored therein. If a comparison occurs, indicating that the tags are identical (a hit), a signal is sent via line 39 to cache controller 40.

While cache controller 40 and other operational and control entities, are, for illustration purposes, shown as independent entities, it should be understood that their functions, to be hereinafter described, may be incorporated into a CPU and not appear as totally independent entities.

Cables 42 and 44 provide to cache controller 40 the cache address bits from cache bus 24 and the complete physical address from register 20, respectively. Lines 43

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 14 of 21

and 48 respectively provide the means for setting and reading out to cache controller 40, the dirty and shared bits respectively. Cache controller 40 also provides signals via cable 42 to cache bus 24 which control the read in and read out of data from cache data store 26, cache tag store 28, dirty bit register 30, shared bit register 32 and bus monitor cache tag store 34.

Communications between the main elements of the system shown in FIG. 2 occur on system bus 42. In actuality, system bus 42 is comprised of 3 separate bus systems, i.e., data bus 182, address bus 184 and control bus 186. Control bus 186 is in turn comprised of main control lines 53, shared line 58 (which is connected to all cache controllers), bus lock line 59 (which is monitored by all units connected to system bus 42) and hold lines 57 and 57'. Hold line 57 emanates from cache controller 40 and hold line 57' emanates from cache controller 40. Both hold lines 57 and 57' are monitored by all units connected to system bus 42.

Each hold line 57, 57' and lock line 59 performs somewhat similar functions (i.e. exclusion of others from the system bus). When a cache controller asserts its hold line, all other potential users of the system bus are kept off the bus until the hold line is released. For instance (as will hereinafter be described in detail), it will sometimes be the case that a cache controller, in monitoring the system bus 42, will detect that a more recent version of the data being returned to another requestor is also present in its own cache store. In such case the monitoring cache controller will set its hold line which indicates to the data requestor, that more data may be coming and not to release its connection to the system bus. Thus the hold line effectively prevents any subsequent independent transaction from occurring until the data requestor is assured of having received the most updated data available.

With respect to bus lock line 59, an entity connected to the bus must request access to it and is granted that access only after the system ensures that no other entity having higher priority is queued up and waiting with a similar request. Once an entity is granted access to bus lock line 59 and asserts it, only the asserting entity and memory control unit 54 have access to the system bus. Thus when the asserting entity issues a read request, memory control unit 54 can respond without fear of any other entity having an opportunity to access the main memory and to change the requested data. When the particular operation or operations are concluded, bus lock line 59 is released and becomes available to any other requesting entity.

Data bus 182 carries data between all main subsystems (e.g. between the cache memory systems and main memory 14). Each cache memory is provided with a data bus interface 50 which provides access to and from data bus 182. In a like manner, address bus 184 is accessed by each cache memory via an address bus interface 52. Control bus 186 is connected via cable 49 to cache controller 40 and provides the necessary traffic pathways for various system control signals. Each of busses 182, 184 and 186 is also connected to memory control unit 54 which manages all main memory operations. Each of busses 182, 184 and 186 further connected to I/O interfaces 61 which control all transfers between connected input/output devices and the various system busses; and to system control interface unit 63 which provides system initialization control and diagnostics.

An important portion of each cache memory for assuring data integrity, is bus monitor cache tag store 34. Bus monitor cache tag store 34 is identical in structure to cache tag store 28 and contains an identical set of tags as are contained in cache tag store 28.

When address bus interface 52 monitors a physical address on address bus 184, it transmits that address via cable 55 to external address register 58. Output cable 60 from external address register 58 carries the cache address portion of the monitored physical address and causes the memory line corresponding thereto in bus monitor cache tag store 34 to be read out via cable 62 to comparator 64. The other input to comparator 64 occurs on cable 66 and is the tag portion of the physical address stored in external address register 58. An equivalency indication from comparator 64 is fed via lines 67 to cache controller 40. Another input to cache controller 40 is provided via cables 86 and 88 which carry the full physical address monitored by address bus interface 52. Cache controller 40 controls the operations of address bus interface 52 and data bus interface 50 via signals emplaced on lines 70 and 72. Data bus interface 50 operates in much the same manner as address bus interface 52, however it captures data appearing on data bus 182 and provides it upon command via cable 51 to cache data store 26.

Referring now to FIG. 3 (in conjunction with FIG. 2), a decision tree is illustrated showing operations which occur when CPU 18 issues a read request to main memory 14. Under such circumstances, CPU 18 issues its read request via cable 80 to cache controller 40. Previously, the physical address of the data requested to be read from memory was inserted by CPU 18 into physical address register 20. Cache controller 40 then causes the tag portion of physical address register 20 to be compared against the tags stored in cache tag register 28. If a match is found by comparator 36, a signal is provided by line 38 to cache controller 40 indicating "hit". If cache controller 40 detects no "hit" signal within a predetermined time period, it assumes that a "miss" signal has been generated and that corresponding data is not present in cache data store 26. Similarly, the cache address residing in physical address register 20 causes the dirty and shared status bits to be read out via lines 43 and 45 to cache controller 40. Thusly, cache controller 40 knows whether the data being sought is present in cache data store 26 (hit or miss); whether it is dirty or not; and whether it is shared or not.

As shown in FIG. 3, the top four branches of the decision tree all assume that the data being sought is present in cache data store 26. In such case, it is irrelevant whether it is dirty or not, or shared or not shared and cache controller 40 causes cache data store 26 to read out the requested data via cable 27 to CPU 18.

If, on the other hand, cache controller 40 detects that the requested data is not in cache data store 26 (a miss), and that the data stored at the addressed location in cache data store 26 is both dirty and shared, a series of operations are then undertaken. Initially, cache controller 40 issues a main memory read request to memory control unit 54 indicating data is required from the physical address location stored in physical address register 20. Memory control unit 54 then accesses the requested data from the physical address in main memory 14 and provides it via data bus 182 to data bus interface 50. Before that data may be written into cache data store 26, the unwanted data presently stored at that address within cache data store 26 must be saved. Since the unwanted data is "dirty", it is written back into main memory 14 via cable 51 and data bus interface 50. In

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 15 of 21

that way, the data in main memory 14 is updated to correspond to the unwanted dirty data presently residing in the addressed position of cache data store 26. The dirty bit corresponding to that address is then reset to zero and data bus interface 50 is commanded to write the requested data (now held in data bus interface 50) into cache data store 26 at the cache address indicated by physical address 28. Simultaneously, the tag portion of the requested data is entered into cache tag store 28 and bus monitor cache tag store 54. The shared bit corresponding thereto in shared bit store 32 is then updated in accordance with the "bus shared" signal appearing on control bus 106. (the operation of which will be described hereinbelow.) Subsequently, the data newly entered into cache data store 26 is read back to CPU 10 via cable 27.

Each cache controller continuously monitors system bus 42 for two types of operations: (a) a data write to main memory 14 and (b) a data return from main memory 14 to fill a *read request*. The monitoring is independent of where the read or write request originates (e.g. a cache/CPU, a system bus interface, a system control interface, an I/O interface). It is this continuous monitoring which greatly assists the system to maintain data integrity.

Returning to FIG. 2, if it is assumed that cache controller 40 issues a data *read* signal onto control bus 106, that signal is sensed by cache controller 40' which instructs address bus interface 52, to latch the address to be subsequently placed onto address bus 46. Address bus interface 52' latches the requested physical address from the address bus and places it into external address register 58'. There, its tag portion is utilized to determine if bus monitor cache tag store 54' has a tag which compares with the address tag portion just inserted into external address register 58'. If a compare occurs, cache controller 40' sets a "hold" signal onto its hold line in control bus 106 which, in essence, tells cache controller 40 that more data is potentially on the way and not to release its system bus connection. Cache controller 40' further determines whether the corresponding data in cache data store 26' is dirty or not. If it is found to be dirty, it then causes the addressed "dirty" data in cache data store 26' to be placed onto data bus 103. Data bus interface 50 captures that data and feeds it to cache data store 26 for storage. Additionally, cache controller 40' sets a "bus shared" line on control bus 106 which indicates that the newly stored data in cache data store 26 is present in more than one location. (Obviously, if cache controller 40' had found that its data was not dirty, then no transmission to cache data store 26 would have been necessary as the data in cache data store 26' would have been identical to that being read from main memory 14).

When cache controller 40' sets the "bus shared" line to the one state, cache controller 40 also sets a shared bit in shared register 32 (which corresponds to the data just read into cache data store 26) to the one state.

Assuming now that data is to be read from a cache data store into main memory 14, such a data transfer is, as aforestated, monitored by non-associated cache controllers. (i.e., those controllers which control cache data stores other than the cache data store from which data is being read). Thus, if cache controller 40' senses a write command generated by cache controller 40 onto control bus 48, it instructs address bus interface 52' and data bus interface 50' to latch the address and data being written to main memory 14. Subsequently, a tag comparison is performed to determine if any of the tags in bus monitor cache tag store 54' are equal to the tag stored in address bus interface 52'. If so, then cache controller 40' instructs data bus interface 50' to update cache data store 26' with the new data.

Returning now to FIG. 3, the decision tree shown therein indicates that on a CPU read request, if a "cache miss", "dirty" and "not shared" sequence of findings occur, the operation is the same as when "shared" data is found. In other words whether the data is shared or not shared is irrelevant to the sequence of operations.

If however, a cache miss occurs and the data is found to be not dirty, (and shared or not shared) the sequence of operations is altered. In such a case, a main memory read request is generated by cache controller 40 to memory control unit 34 and the main memory data is written into cache data store 26. Since data at the addressed position of cache data store 26 is "not dirty" there is no need to update the identical data appearing in main memory 14. The "bus shared" line is then sampled and the shared bit is set in accordance with whether another cache controller, in response to the read request appearing on the control bus, has set the shared line to indicate that it too has the same data. Finally, the contents of cache data store 26 are read to CPU 10.

Turning now to FIG. 4, the decision tree shown therein will be helpful in understanding the operation of the system when a data return indication is monitored on the system bus by a non-requesting cache controller. As above stated, each cache controller monitors the control bus for both data reads from main memory and writes to main memory. In the instance shown in FIG. 4, it is assumed that a read from main memory 14 has been requested by cache controller 40 and that a data return response thereto has been monitored by cache controller 40'. As aforestated, cache controller 40' causes the physical address of the data being accessed to be latched in address bus interface 52'. It then determines if a cache "hit" occurs; whether the data is dirty or not dirty and whether it is shared or not shared. If the conditions hit, dirty and shared are found, cache controller 40' sets its hold line on control bus 106 and instructs cache data store 26' to write its stored data onto data bus 103 where it is picked up and stored in cache data store 26 by data bus interface 50. Cache controller 40' also sets the shared line on control bus 106 equal to one, assuming the requestor of the data is a central processing unit.

If in the case discussed above, cache controller 40' finds that the data indication in shared bit register 32' is "not shared", the process is much the same as in the shared condition however shared bit register 32' is set to one to properly indicate the shared state of the data in cache data store 26'.

If a cache hit is found but the data in cache data store 26' is found to be "not dirty", the following actions occur. If the data is found to be shared, cache controller 40' sets the bus shared line to a one, assuming the requestor is a central processing unit. If the data is found to be "not shared" the bus shared signal is also set to a one and the shared bit associated with the respective data line in cache data store 26' is set to one indicating that the data stored therein is, in fact, shared. Here again, it is assumed the requestor is a central processing unit. As is obvious, if there is a cache miss in this procedure, no actions are required.

The timing diagrams of FIGS. 18a, 18b and 18c illustrate various types of bus transactions.

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™ a product of Patent Logistics, LLC - Page 18 of 21

FIG. 18e shows a byte/word/double write immediately followed by the Command-ID and address portion of a double (64 bit) read followed by an MCU data return of the requested double word.

FIG. 18f demonstrates the use of both the lock and hold lines for a cache fetch/write back. That case corresponds to when a cache fetch is issued, a miss occurs and dirty data is resident in the addressed position. Thus the dirty data must be written back to main memory prior to the requested data from main memory being written into the cache store. The associated cache controller is shown asserting both its hold line and the lock line and sending command ID to MCU 54 for an octal word read. The asserted hold line prevents MCU 54 from writing into the cache store until it has been released and the asserted lock line prevents another bus connection from using the bus during this sequence. Subsequent to the write back, and release of both the hold and lock lines, MCU 54 is free to write the requested four double words to the cache store.

FIG. 18g demonstrates another use of the hold line. A bus connection is shown requesting a double word read and the MCU 54 returning the requested double word. The caches latch the address of the double word, and do directory look-ups in the following cycle. If a dirty match is found by a cache, that cache asserts hold shortly before the end of the cycle. The hold line prevents other connections from using the bus until the write-back cache re-transmits the double word along with its address and thereafter releases. The retransmission is indicated by the assertion of a bus valid signal on the system bus.

Turning now to FIG. 5, the procedures followed when a write to memory command is received from a CPU will be described. In such a case, the initial operation is much the same as when a read request is received, i.e., the physical address accompanying the write command is analyzed to determine whether the data sought to be written is either present or not present in cache data store 26 (a hit or a miss). If a hit occurs and it is found that the data is dirty and shared, cache controller 40 causes cache data store 26 and cache tag store 28 to store the respective data and tag bits. However, since it was found that the CPU is writing to a shared location, invariably, a write through action occurs to main memory 14. Thus the data, in addition to being written into cache data store 26, is also written to main memory 14 via a write command placed on control bus 106.

Assuming the shared data is still resident in cache data store 26', the write through is sensed by cache controller 40' which determines that its cache data store 26' includes data having the same tag. Cache controller 40' then instructs data bus interface 50' to update that data position with the data being written from cache data store 26 to main memory 14. (which has been captured by address bus and data bus interfaces 52' and 50' respectively). This ensures that each of the cache data stores has the most updated information. Since the main memory data and the cache data stores now contain the same information, the dirty bit is reset to 0.

If it is found that the data being written to main memory is not shared, all that is required is that the cache data store 26 be updated. Note that the dirty bit remains equal to one as the data still is not the same as that stored in main memory 14, as no write through to main memory has occurred.

If there is a cache hit and not dirty and shared indications are found, the operation is identical to that for the "cache hit", "dirty", and "shared". If the data is found to be not shared, then the cache is updated and the dirty bit is again set to one as the data in cache data store 26 differs from the data stored at the same memory position in main memory 14.

If a CPU write to memory command is received and a cache miss is found but with a dirty indication for the data already stored at the particular addressed position within cache data store 26, then a considerably more complex set of steps take place. This is irrespective of whether the data is shared or not shared. First, the main memory read request is transmitted to memory control unit 54 by cache controller 40. Then, the dirty entry in cache data store 26 is written into main memory 14. The actual data being sought to be modified is written from main memory 14 back into cache data store 26 and the cache tag store and bus monitor cache tag store are updated. The new data on cache data store 26 is then updated by the act of CPU 18 overwriting new data in the same data position. (It should be recalled that the system retains the capability to selectively overwrite a full double word, single word or any combination of bytes.) Since a write to memory signal was sensed on control bus 106, if another cache controller finds that it too has, in its associated cache data store, the data being accessed, it will set the "bus shared" line on control bus 106 indicating shared data. The bus shared signal is sensed by cache controller 40 and the shared bit in shared bit register 32 is set to one. Furthermore, the data recently written into cache data store 26 by CPU 18 is also written through into main memory 14 and thus is sensed by cache controller 40' so that it may modify its associated cache data store. The dirty bit is then reset to 0 in dirty bit register 30. Obviously, if the bus shared signal is equal to 0, there is no need to write through the CPU entry to main memory 14. As shown by the lower branches of the decision tree in FIG. 5, the operations are the same whether the data is initially indicated as being shared or not shared.

If, in response to a CPU write signal, cache miss, not dirty and shared (or not shared) indications are found, the operation is much the same as with the "dirty" instance, however, there is no need to update main memory with the "not dirty" entry as it contains the same data.

Turning now to FIG. 6, the decision tree shown therein illustrates the various system actions in response to a write to memory being sensed on the system bus. If the non-requesting CPU/cache system finds that the data being written to memory from another CPU/cache system or I/O interface is found in its associated cache data store, and it is indicated as being shared, then it merely updates its cache to coincide with the new data being written to memory. In the not shared cases indicated for a cache hit, there is an error state sensed if the same data is found in a cache data store which is being written to memory from another cache data store. In all cases, the cache data store is updated in accordance with the data being written to memory.

What has been discussed to this point has involved, in the main, double word accesses from memory and double word writes to memory. As illustrated in FIG. 1 however, the data format employed by the system shown in FIG. 2 includes two, 4 byte words per line of cache data storage. In many instances, it is not only of interest to read or write both data words, but it is also

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC

beneficial to read or write multiples of double words in a single operation without having to perform individual stores or writes for each double word. This is particularly useful when accessing or writing instruction sets or operating on long character strings.

To handle multi-double word operations, a status register is provided. Thus, as shown in FIG. 2, each of cache controllers 40 and 40' is provided with multi word status registers 90 and 90' respectively. These status registers are employed during multi word transfers.

Referring to FIG. 7, a flow diagram is illustrated which illustrates the occurrences when a multi word data transfer appears on system bus 42 and is monitored by a cache controller other than the one which issued the multi word read request. Each cache controller continuously monitors the system bus for a multi word transfer. During a data return, a cache controller may detect that a data word being transferred has the same tag as a tag stored in its associated bus monitor cache tag store. The cache controller sets in its associated multi word transfer register a one bit in a position which corresponds to the address of the cache tag which evidenced the hit indication. Similarly the cache controller upon detecting the first "hit" in multi-word transfer, sets its bus hold control line which notifies the cache controller receiving the data that it should expect additional data.

The cache controller continues to monitor the system bus until the multi-word transfer has ended. At such time (assuming that cache controller 40' has been monitoring the bus), multi word register 90' has stored therein a series of zeros and ones, with the ones in positions corresponding to addresses of the cache tags where a hit indication occurred.

The cache controller then addresses its tag store at the address indicated by the first hit. If the data stored at that address is indicated as dirty, it is transmitted to the data requestor. If the data is found to be not dirty, the next address is accessed and the process repeated until all addresses corresponding to positions of the multi-word transfer register with one bits, have been examined.

Subsequently, cache controller 40' causes the data which is dirty and corresponds to the ones in multi word transfer register 90', to be transmitted to the interface units associated with the cache controller 40 for storage in cache data store 26.

Turning now to FIG. 8, a flow diagram illustrating an N multi-word data read from memory is illustrated. Initially, a CPU requests a multi word data read from its associated cache, e.g. CPU 10 issues to cache controller 40 via line 80 a multi word data request. However, the initial cache tag requested is found not to be contained in cache data store 26 (a miss). Controller 40 then issues a multi word instruction request to memory control unit 54. Before the actual data transfer occurs the memory position to receive the first double word in cache tag store 26 is addressed. The dirty bit corresponding to the data presently at that address is examined to see if it is set to a one. If it is not, it does not need to be written back to main memory and a corresponding status bit in multi word transfer register remains at 0 indicating that that position may be overwritten. If it is set to one, it must be written back to main memory for update purposes before it may be overwritten.

After either a write back or a finding that a write back is not necessary, the cache tag store is addressed. If there is a hit, a corresponding status bit in multi word transfer register 90 is set equal to one. Subsequently, until N tag store addresses have occurred, the process repeats itself until all N double words have been examined. Then, controller 40 allows a data return from memory control unit 54 which reads out the N words onto data bus 182 where they are captured by data bus interface 58. Multi word transfer register 90 then comes into play and controls which positions in cache data store 26 are updated. (I.e., only positions corresponding to the positions of multi word register 90 which are set to 0). In this manner, not only is the dirty data written back to main memory and main memory completely updated, but also that data received from main memory 14, which is not most current, is inhibited from being stored in the cache data store.

Under certain circumstances, conflicts may occur between cache memories. One such conflict may arise during a data return from main memory to a requestor which is monitored by two or more caches. If those caches register hits for the data unit (or units) being returned from main memory, each will set its respective hold line before it has had a chance to examine the dirty status for data at the monitored address. To prevent a conflict, the procedure shown in FIG. 9 is employed. In sum, each cache controller determines if the others hold line was set on a previous clock cycle. If so the earlier set hold lines dominates. If both hold lines were set during the same clock cycle, then the cache controller associated with the CPU bearing the lower identification number takes precedence.

It is to be understood that the above described embodiments of the invention are illustrative only and that modifications throughout may occur to those skilled in the art. Accordingly, this invention is not to be regarded as limited to the embodiments disclosed herein, but is to be limited as defined by the appended claims.

What is claimed is:

1. A data processing system including one or more central processing units, main memory means, and bus means, for each central processing unit the invention comprising:

cache memory means coupled between the central processing unit and said bus means;

bus monitor means associated with said cache memory means and coupled to said bus means for detecting on said bus means an address associated with a data unit transferred from said main memory means to a bus connection requesting the data unit;

means coupled to said cache memory means and to said bus means for determining if data having the same address as said transferred data unit is present in said cache memory means and, if present, for asserting a hold signal on said bus means, the assertion of the hold signal indicating at least to the bus connection requesting the data unit that another data unit may be transmitted over said bus means; and

means for detecting whether data corresponding to the address of said transferred data unit and determined to be stored in said cache memory means may be different in content from said transferred data unit and, if so, transmitting said data from said cache memory means to said bus means for reception by the bus connection requesting the data unit.

2. The invention as defined in claim 1 wherein said bus connection requesting the data unit, in response to sensing the assertion of said hold signal, maintains an

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 18 of 21

interaction with said bus means so that the bus connection requesting the data unit may receive said different content data from said cache memory means.

3. The invention as defined in claim 2 and further comprising:

a multi-data unit transfer register; and

means coupled to said bus means and responsive thereto for detecting a multi-data unit read onto said bus means from said main memory means and for determining if any said data unit is present in said cache memory means; said detecting means further including means for recording an indication of such presence in said multi-data unit transfer register and causing said hold line to be asserted.

4. The invention as defined in claim 3 and further including means for transferring from said cache memory means to the recipient of said multi-data unit transfer each said data unit indicated by said multi-data unit transfer register as being present and which is also determined to be dirty.

5. A data processing system including at least first and second central processing units, main memory means, and bus means, the invention comprising:

at least first and second cache memory means respectively coupled between said first and second central processing units and said bus means;

bus monitor means associated with each said cache memory means respectively and coupled to said bus means for detecting on said bus means an address associated with a data unit being read from said main memory means;

means coupled to said associated cache memory means and to said bus means for determining if said corresponding to said address as detected by said bus monitor means is present in the associated cache memory means and, if said data is found to be present, for asserting a first hold signal on said bus means, the assertion of the first hold signal indicating at least to a bus connection reading the data unit that another data unit may be transmitted over said bus means; and

means for detecting whether the data corresponding to said address and stored in said associated cache memory means may be different in content from that detected as being read and, if the data is determined to possibly be different in content, transmitting said data from said associated cache memory means to the bus connection reading the data unit from said main memory.

6. The invention as claimed in claim 5 wherein each said cache memory means comprises a cache data store, a cache tag store, a shared bit store and bus monitor cache tag store.

7. The invention as claimed in claim 6 wherein said bus monitor means captures the physical address of said data from said bus means and provides the cache address and cache tag portions thereof to said determining means.

8. The invention as claimed in claim 7 wherein said determining means compares said cache tag from said bus monitor means with a cache tag stored in its associated bus monitor cache tag store at said cache address provided by said bus monitoring means to determine if said cache tags are identical.

9. The invention as claimed in claim 8, wherein said detecting means includes a dirty bit store.

10. The invention as defined in claim 5 and further comprising for each said cache memory means:

a multi-data unit transfer register; and

means coupled to said bus means and responsive thereto for detecting a multi-data unit read on said bus means from said main memory means and for determining if any said data unit is present in said cache memory means; said detecting means further including means for recording an indication of such presence in said multi-data unit transfer register and causing said hold line to be asserted.

11. The invention as defined in claim 10 and further including means for transferring from said associated cache memory means to the recipient of said multi-data unit transfer each said data unit indicated by said associated multi-data unit transfer register as being present and which is also determined to be dirty.

12. The invention as defined in claim 5 wherein said determining means includes means coupled to said bus means for monitoring said bus means for sensing the presence of the assertion of a second hold signal thereon and for determining priority between the first hold signal and the second hold signal.

13. The invention as defined in claim 12 wherein said detecting means is coupled to said monitoring means and is responsive thereto for transmitting said data from said associated cache memory means in accordance with the priority determination of said monitoring means.

14. In a data processing system comprising at least one central processing unit having a cache memory coupled to a system memory through a system bus, the system bus including signal lines for conveying, during bus cycles, address information and data units associated with addresses between the system memory and other bus connections coupled to the system bus, the cache memory comprising:

means for monitoring bus transactions occurring on the system bus for determining if a bus transaction occurring during a current bus cycle is an operation initiated by another bus connection that reads one or more data units from an address of the system memory;

means for detecting if the address is associated with one or more data units already stored within the cache memory; and

means having an input coupled to said detecting means and responsive thereto and further having an output coupled to the system bus for asserting thereon during the current bus cycle a HOLD signal line, the assertion of the HOLD signal line indicating to the bus connection initiating the read operation that at least one of the data units associated with the address is stored within the cache memory and that the cache memory will, if the data unit is determined to be marked as dirty within the cache memory, transmit the stored data unit to the bus connection during a subsequent bus cycle.

15. A cache memory as set forth in claim 14 and further comprising:

a multi-data unit transfer register; and

means coupled to the system bus and responsive thereto for detecting an occurrence of a multi-data unit read operation and for determining if any data unit read by the multi-data unit read operation is stored in the cache memory, the multi-data unit read operation detecting means further including means for recording in the multi-data unit transfer register an indication that the data unit is stored within the cache memory and further comprising

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 19 of 21

means, coupled to the asserting means, for causing the HOLD signal line to be asserted.

16. A cache memory as set forth in claim 14 wherein each cache memory is coupled to an associated HOLD signal line for asserting same and wherein said monitoring means includes means for sensing an assertion of another HOLD signal line and further includes means for determining priority between an assertion of the associated HOLD signal line and an assertion of another HOLD signal line.

* * * * *

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 20 of 21

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 4,939,641

DATED : July 3, 1990

INVENTOR(S) : Schwartz et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Claim 5, Col. 13, line 33, delete "said" (second occurrence) and insert --data--.

Signed and Scaled this

Twenty-fifth Day of June, 1991

Attest:

Attesting Officer

HARRY F. MANBECK JR.

Commissioner of Patents and Trademarks

This PDF of U.S. Utility Patent 4939641 provided by Patent Fetcher™, a product of Patent Logistics, LLC - Page 21 of 21

United States Patent [19]

Whipple

[11] Patent Number: 5,077,733

[45] Date of Patent: * Dec. 31, 1991

[54] **PRIORITY APPARATUS HAVING PROGRAMMABLE NODE DWELL TIME**

[75] Inventor: David L. Whipple, Braintree, Mass.

[73] Assignee: Wang Laboratories, Inc., Lowell, Mass.

[*] Notice: The portion of the term of this patent subsequent to May 15, 2007 has been disclaimed.

[21] Appl. No.: 405,792

[22] Filed: Sep. 11, 1989

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 317,100, Feb. 28, 1989, Pat. No. 4,926,419, which is a continuation-in-part of Ser. No. 97,775, Sep. 17, 1987, abandoned, which is a continuation-in-part of Ser. No. 712,692, Mar. 15, 1985, Pat. No. 4,719,622.

[51] Int. Cl.[5] H04L 5/22

[52] U.S. Cl. 370/85.6; 370/85.11; 340/825.5

[58] Field of Search 370/85.1, 85.2, 85.4, 370/85.5, 85.6, 85.9, 85.11; 340/825.05, 825.5, 825.51

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,232,294	11/1980	Burke et al.	
4,320,502	3/1982	deVeer .	
4,336,480	12/1982	Van Hanes .	
4,559,595	12/1985	Boudreau et al. .	
4,583,089	4/1986	Cope .	
4,626,843	12/1986	Szeto et al. .	
4,663,756	5/1987	Retterath .	
4,719,622	1/1988	Whipple et al .	
4,763,122	8/1988	Franaszek .	
4,814,974	3/1989	Narayanan et al.	370/85.6
4,926,419	5/1990	Whipple	370/85.11

OTHER PUBLICATIONS

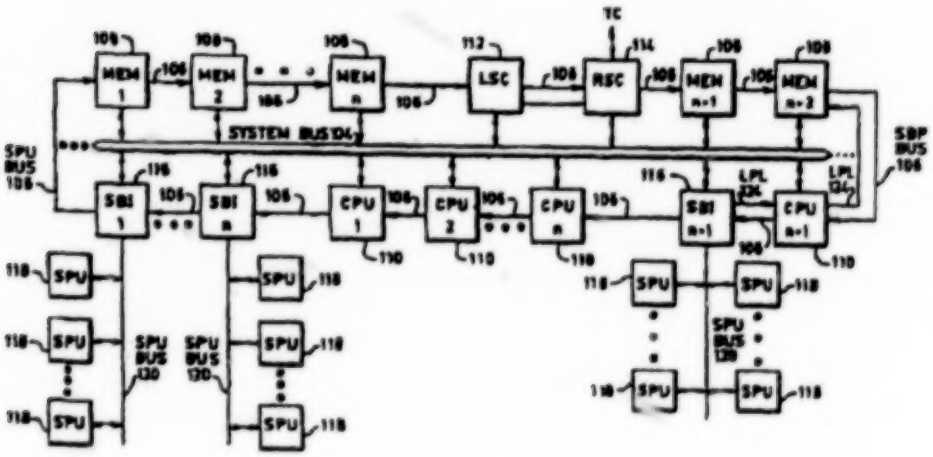
IBM Technical Disclosure Bulletin, vol. 20, No. 2, D. F. Bantz.

Primary Examiner—Douglas W. Olms
Assistant Examiner—Melvin Marcelo
Attorney, Agent, or Firm—Michael H. Shanahan

[57] **ABSTRACT**

Apparatus for determining priority of access to a bus by nodes in a group of nodes attached to the bus. For purposes of determining priority, the apparatus arranges the nodes in a circular configuration and selects one of the nodes as the "anchor node". The anchor node has the highest priority and the priorities of the other nodes are determined by their positions in the circle relative to the anchor node. Each node includes a presettable counter for indicating a number of times that a node may access the bus before the highest priority is rotated to another node. After a device represented by one of the nodes accesses the bus for a predetermined number of speames the current anchor node ceases being the anchor node and the next node in the circle becomes the new anchor node. The priorities of the nodes change to reflect the new location of the anchor node.

19 Claims, 10 Drawing Sheets



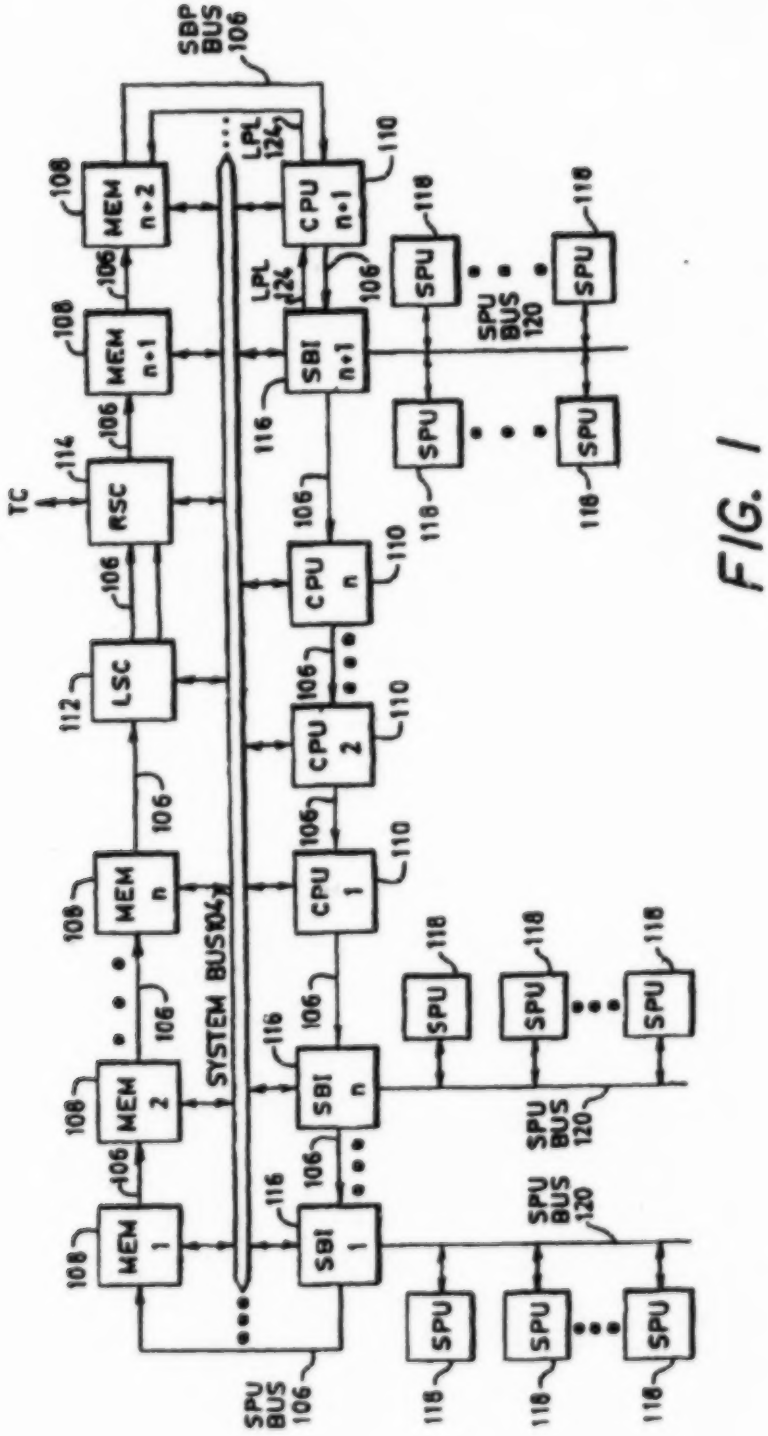


FIG. 1

BEST AVAILABLE COPY

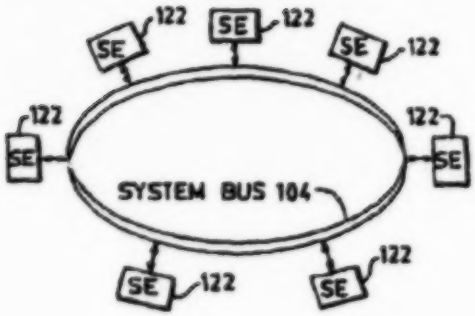


FIG. 1A

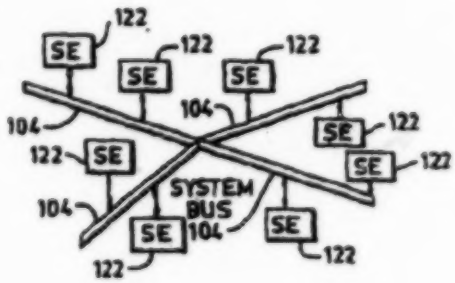


FIG. 1B

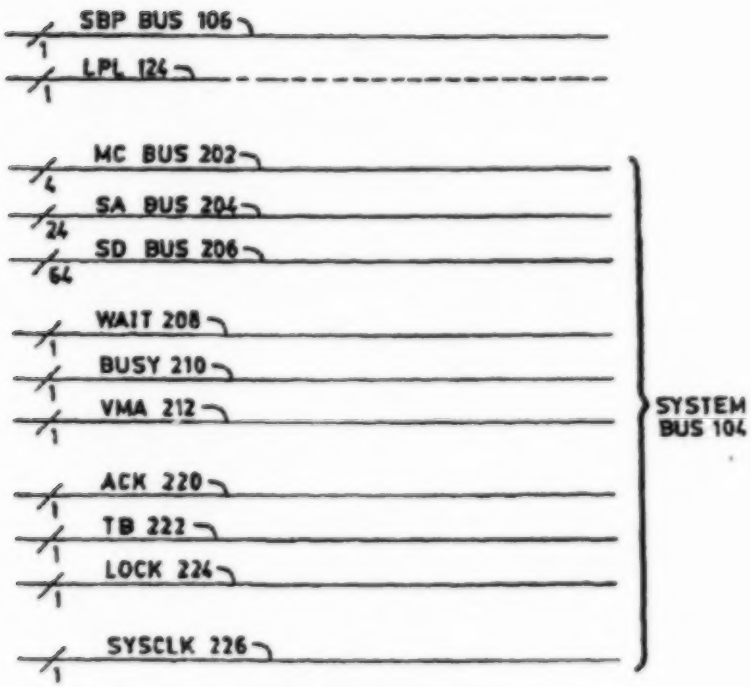


FIG. 2

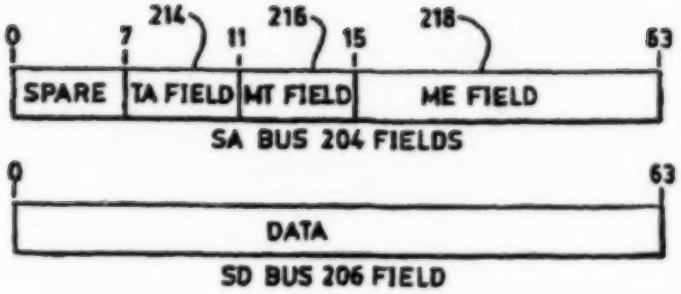

0
7
214
11
216
15
218
63
SPARE
TA FIELD
MT FIELD
ME FIELD
SA BUS 204 FIELDS
0
63
DATA
SD BUS 206 FIELD

FIG. 2A

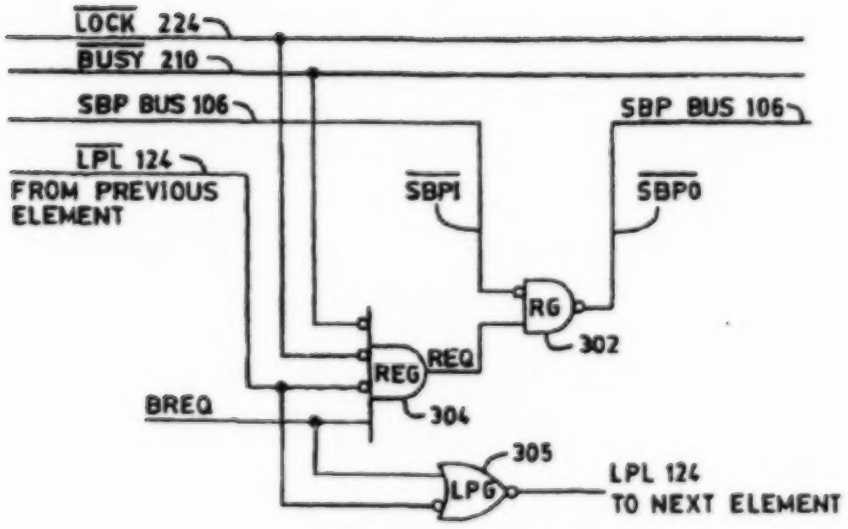

LOCK 224
BUSY 210
SBP BUS 106
SBP BUS 106
LPL 124
FROM PREVIOUS ELEMENT
SBP1
SBP0
RG
302
REG
REQ
304
BREQ
305
LPG
LPL 124
TO NEXT ELEMENT

FIG. 3A

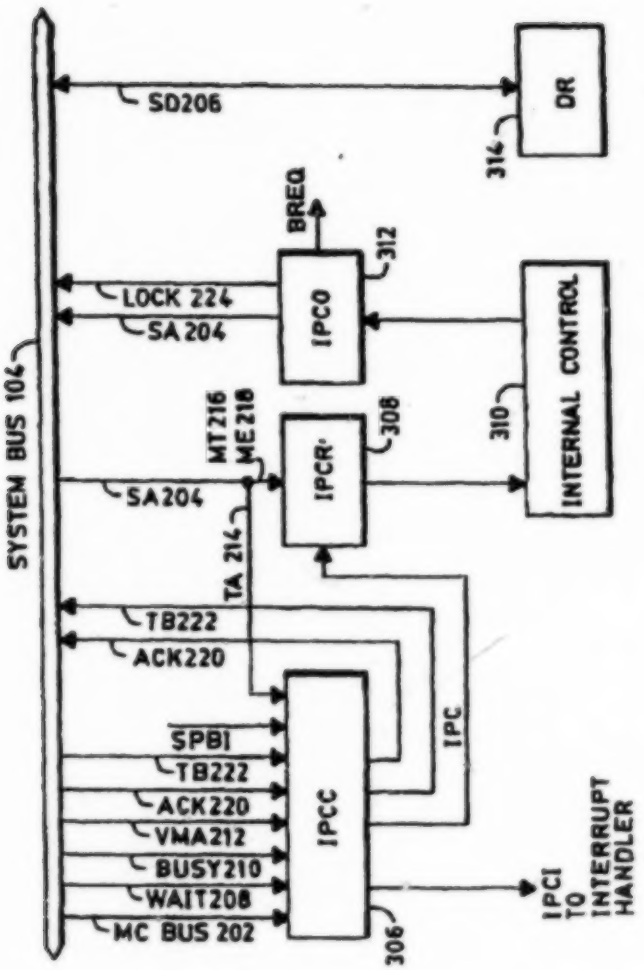

SYSTEM BUS 104
SD206
DR
314
BREQ
312
LOCK 224
SA 204
IPCO
INTERNAL CONTROL
310
MT 216
ME 218
IPCR
308
SA204
TA 214
TB222
ACK220
SPBI
TB222
ACK220
VMA212
BUSY210
WAIT208
MC BUS 202
IPCC
IPC
306
IPCI
TO
INTERRUPT
HANDLER

FIG. 3B

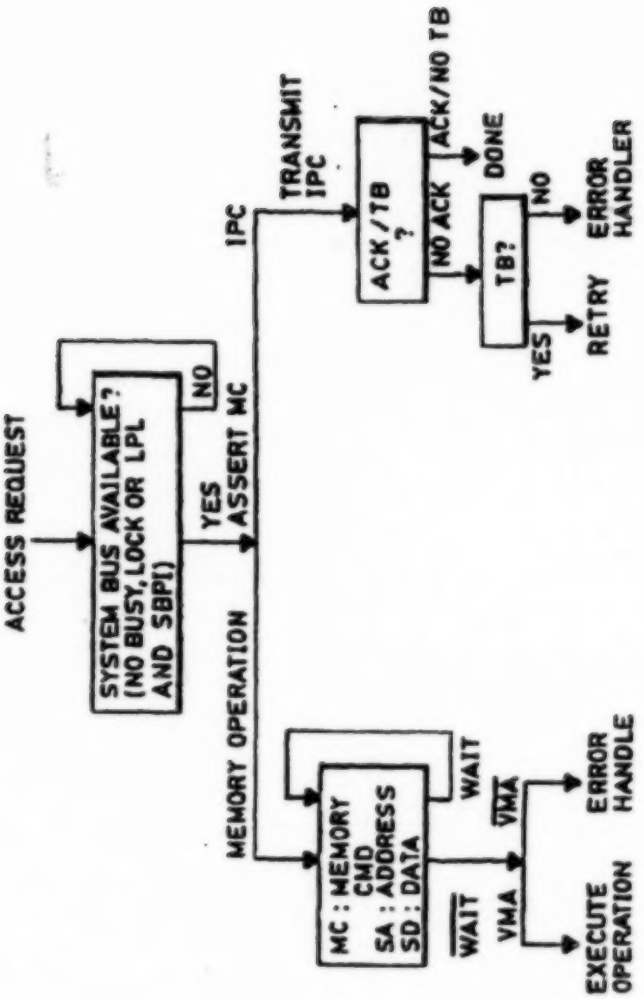

ACCESS REQUEST
SYSTEM BUS AVAILABLE? (NO BUSY, LOCK OR LPL AND SBPI)
NO
YES
ASSERT MC
IPC
TRANSMIT IPC
ACK/TB ?
ACK/NO TB
DONE
NO ACK
TB?
NO
ERROR HANDLER
YES
RETRY
MEMORY OPERATION
MC : MEMORY CMD
SA : ADDRESS
SD : DATA
WAIT
VMA
WAIT
VMA
ERROR HANDLE
EXECUTE OPERATION

FIG. 4A

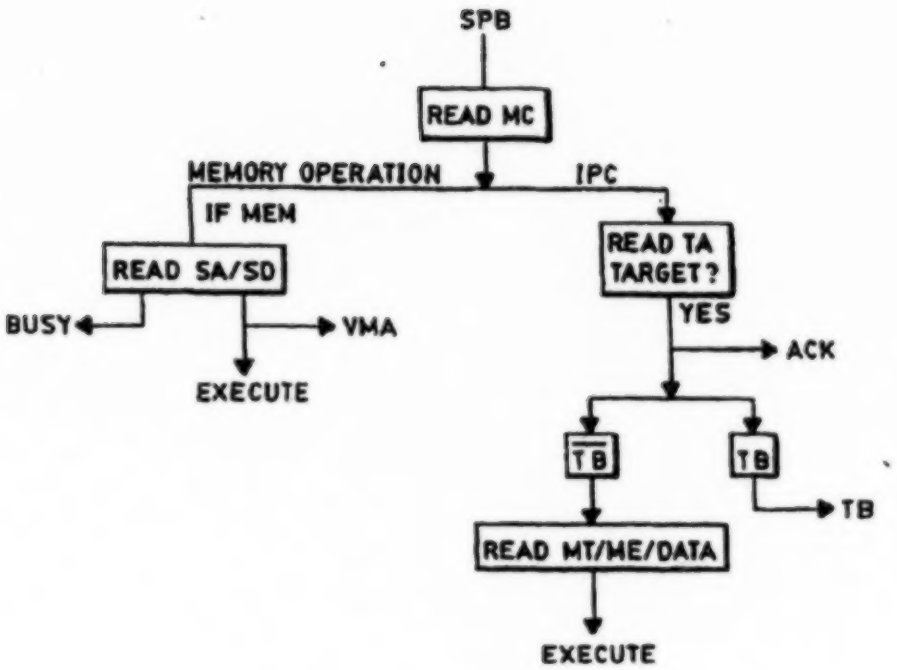

SPB
READ MC
MEMORY OPERATION
IPC
IF MEM
READ SA/SD
READ TA TARGET?
BUSY
VMA
EXECUTE
YES
ACK
TB
TB
TB
READ MT/ME/DATA
EXECUTE

FIG. 4B

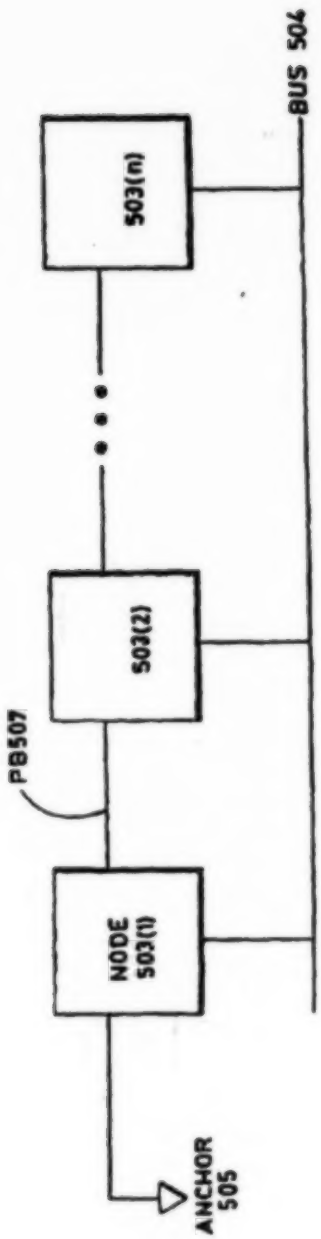


FIG. 5

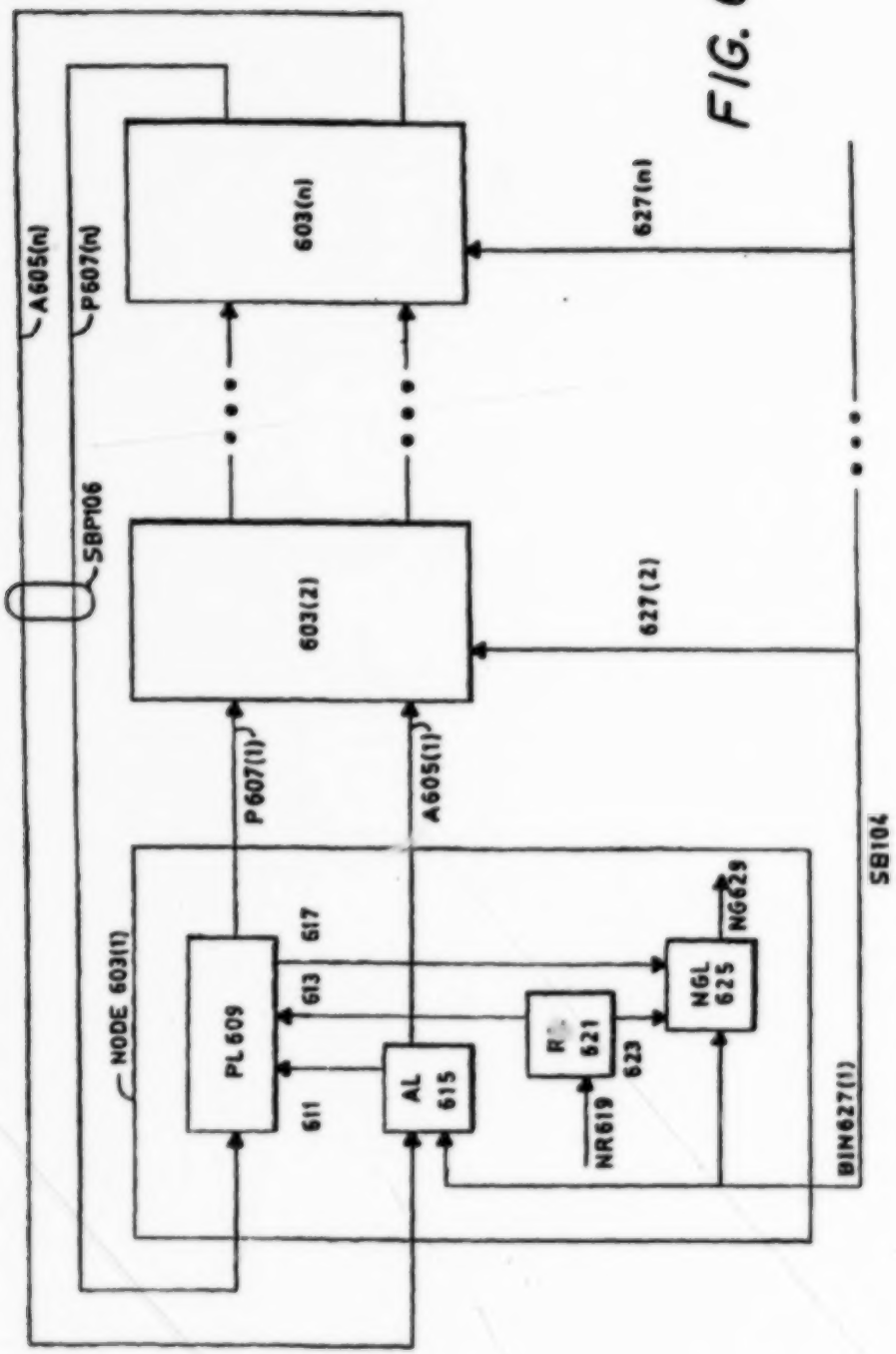

A605(n)
P607(n)
603(n)
627(n)
SBP106
603(2)
627(2)
P607(1)
A605(1)
SB104
NODE 603(1)
PL 609
617
613
611
AL 615
R 621
623
NR619
NGL 625
NG629
BIN627(1)

FIG. 6

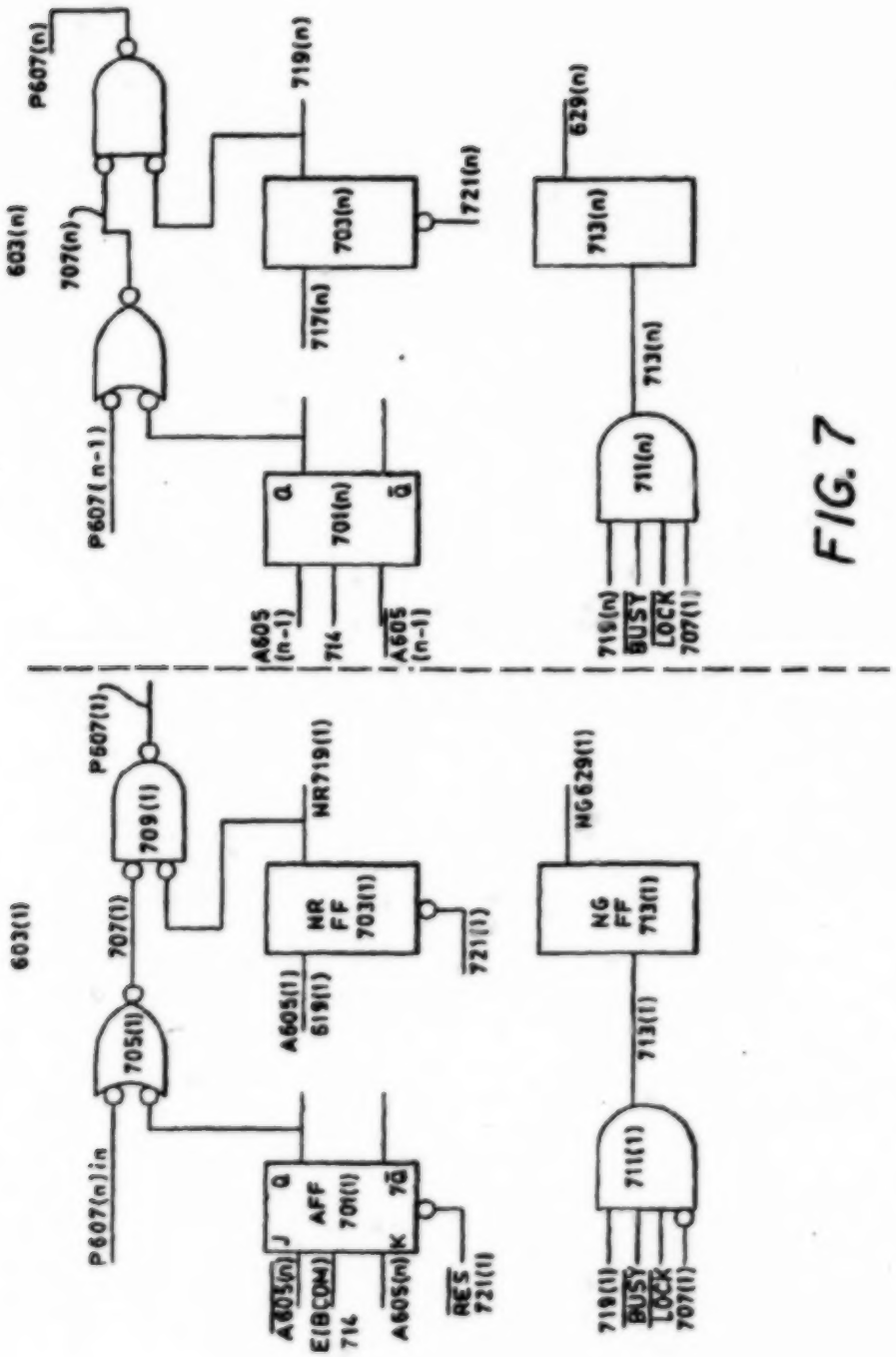

603(n)
P607(n)
707(n)
719(n)
703(n)
721(n)
717(n)
629(n)
713(n)
713(n)
P607(n-1)
701(n)
711(n)
A605 (n-1)
714
A605 (n-1)
715(n)
BUSY
LOCK
707(1)
603(1)
P607(1)
709(1)
NR719(1)
NR FF 703(1)
721(1)
707(1)
705(1)
A605(1) 619(1)
NG629(1)
NG FF 713(1)
713(1)
P607(n)in
AFF 701(1)
711(1)
A605(n)
E(BCON)
714
A605(n)
RES 721(1)
719(1)
BUSY
LOCK
707(1)

FIG. 7

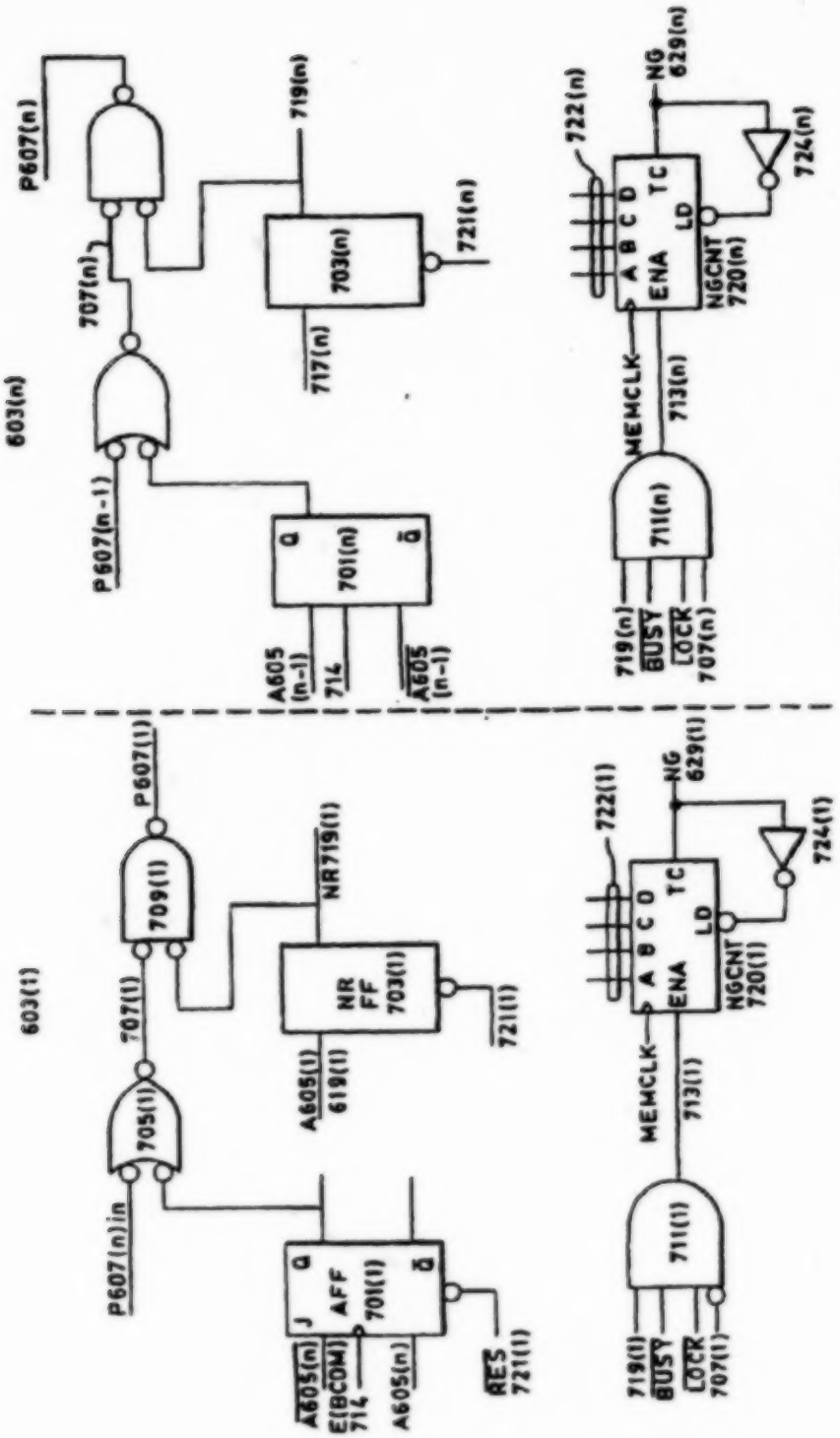

603(n)
P607(n)
707(n)
P607(n-1)
719(n)
703(n)
721(n)
717(n)
701(n)
A605 (n-1)
714
A605 (n-1)
722(n)
NG 629(n)
724(n)
A B C D
ENA
TC
LD
NGCNT 720(n)
MEMCLK
713(n)
711(n)
719(n)
BUSY
LOCK 707(n)
603(1)
P607(1)
709(1)
707(1)
705(1)
NR719(1)
NR FF 703(1)
721(1)
A605(1) 619(1)
P607(n)in
J Q AFF 701(1) Q̄
A605(n) E(BCOM) 714
A605(n)
RES 721(1)
722(1)
NG 629(1)
724(1)
NGCNT 720(1)
713(1)
711(1)
719(1)
BUSY
LOCK 707(1)

FIG. 8

PRIORITY APPARATUS HAVING PROGRAMMABLE NODE DWELL TIME

CROSS REFERENCE TO RELATED APPLICATIONS

The present application is a continuation-in-part of U.S. Ser. No. 07/317,103, Priority Apparatus, filed Feb. 28, 1989 now U.S. Pat. No. 4,926,419 which in turn is a continuation-in-part of U.S. Ser. No. 07/097,773, filed Sept. 17, 1987 (now abandoned) which was a continuation-in-part of U.S. Ser. No. 06/712,492, System Bus Means for Inter-Processor Communication, filed Mar. 15, 1985 and now U.S. Pat. No. 4,719,622, issued Jan. 12, 1988.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to apparatus used in computer systems to determine which of a plurality of devices which share a bus is to have access to the bus at a given instant and more particularly to apparatus in which access is determined by assigning priorities to the device and wherein a device gaining a highest priority to the bus, the anchor node, is granted a programmable number of bus accesses before relinquishing the highest priority to another device.

2. Description of the Prior Art

When a plurality of devices in a computer system share a bus, the computer system must include some way of preventing more than one of the devices from using the bus at once. One technique used in the prior art is to assign each device a static priority; if a device with a high priority and one with a low priority both attempt to access the bus, the device with the higher priority wins. One way of assigning priorities, shown in FIG. 5, is by position. FIG. 5 is a conceptual drawing of a static daisy chain 501. A set of nodes 503, each one of which may provide access to a bus 504 for one or more devices, is arranged in daisy chain fashion, with a priority bus (PB) 507 connecting adjacent members of the chain. The leftmost node 503(1) in the chain has the highest priority, and each succeeding node has a lower priority than the one preceding it. Thus, if node 503(1) and another node 503(*n*) request access simultaneously, node 503(1) receives the access. For purposes of the present discussion, daisy chain 501 may be regarded as having a static "anchor" 505 which marks the point from which all priorities are determined. While static daisy chain 501 successfully determines which of the nodes 503 will have access to the bus at a given time and has the important advantage that the node 503 having priority gains immediate access to bus 504, it does not prevent higher-priority devices from "hogging" bus 504 and denying access to the bus to lower-priority devices. For example, if node 503(1) requests access to the bus on each bus cycle, no other node 503 in chain 501 will ever gain access to the bus 504.

The "hogging" problem may be eliminated if each node is guaranteed a turn at the bus. One way of doing this is disclosed in U.S. Pat. No. 4,342,995, *Data Network Employing a Single Transmission Bus for Overlapping Data Transmission and Acknowledgement Signals*, inventor George T. Shima, issued Aug. 3, 1982. In the system disclosed in that patent, the nodes are daisy-chained together in a loop and a pulse is circulated around the loop. When a node which has a pending bus access request receives the pulse, it gains access to the bus. When the node is finished, it provides the pulse to the next node. If the node which receives the pulse does not wish to access the bus, it holds the pulse for a short time and then passes it to the next node in the chain. While the system of Shima guarantees that none of the nodes will "hog" the bus, it is extremely inefficient in a situation in which one of the nodes uses the bus much more frequently than the other nodes. Once a node in the system of Shima has finished using the bus, it must always wait to begin its next bus access until the pulse has been passed to all of the other nodes on the chain, even though none of the other nodes has a bus access request pending.

What is needed, and what is provided by the invention which is the subject of the present application, is bus priority apparatus which neither permits a single node to "hog" the bus nor requires a node to wait to gain access to the bus when no other node has a request pending. Furthermore, the invention provides that once a node gains the highest bus priority that it maintains the highest priority over the other bus nodes for a programmable number of bus access cycles before relinquishing the highest priority to another node.

SUMMARY OF THE INVENTION

The present invention concerns dynamic circular bus priority apparatus in which each of the nodes connected to the bus has a present priority which is determined by its position in a circular configuration of the nodes relative to one of the nodes which is currently the "anchor node" of the circle. When more than one of the nodes attempts to access the bus simultaneously, the node which gains access is determined by the present priority of the nodes. Furthermore, once a node gains the anchor node priority it retains the highest priority for a programmable number of bus accesses before relinquishing the highest priority to another node. After the programmable number of bus accesses a node other than the current anchor node becomes the anchor node, and the priorities of the nodes are determined by their positions relative to the new anchor node. The invention thus permits the node which presently has priority immediate access to the bus after the expiration of the programmable number of accesses allocated to the node previously having priority and at the same time prevents any node from hogging the bus, and thereby overcomes the problems of the prior art mentioned above.

In a presently-preferred embodiment, each node becomes the present anchor node in sequence. Each time a node gains the anchor position of the bus the node retains the anchor position for a programmable number of bus access cycles made by the node afterwhich the node following the current anchor node becomes the new anchor node. In other embodiments, the next anchor node may be determined by the last node to have gained access to the bus. In such an embodiment, the node following the last node to have gained access becomes the new anchor node, thus guaranteeing that the node which last accessed the bus has the lowest priority for the next access.

It is thus an object of the invention to provide an improved digital computer system.

It is a further object of the invention to provide improved apparatus for determining priority of access to a bus.

It is another object of the invention to provide apparatus for determining priority of access to a bus in

which the node which presently has priority can access the bus immediately but no node can "hog" the bus.

It is an additional object of the invention to provide priority apparatus in which priority is determined by the position of a node relative to a current anchor node whose position in the system changes with each access to the bus.

It is a still further object of the invention to provide priority apparatus in which a node gains a highest bus priority for a predetermined number of bus access cycles before relinquishing the highest priority to another node.

Other objects, advantages and features of the present invention will be understood by those of ordinary skill in the art after referring to the following detailed description of the preferred embodiment and drawings, wherein:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a system incorporating the present invention;

FIGS. 1A and 1B are alternate system bus topologies;

FIG. 2 is a diagram of the bus structure of the present invention;

FIG. 2A is a diagrammic representation of interprocessor communications transmitted through the bus structure of the present invention;

FIGS. 3A and 3B are schematic representations of the system bus interfaces incorporated into the elements of the present system;

FIGS. 4A and 4B are flow charts illustrating the operation of the present invention.

FIG. 5 is a block diagram of a prior-art static daisy chain;

FIG. 6 is a block diagram of the rotating circular chain of the present invention;

FIG. 7 is a logic diagram of an implementation of the present invention in a node; and

FIG. 8 is a logic diagram of a presently preferred embodiment of the invention.

DESCRIPTION OF A PREFERRED EMBODIMENT

Parts A through F of the following Description of a Preferred Embodiment describe the system in which the priority apparatus of the present invention is implemented. The priority apparatus itself is more nearly described in Parts G and H. A presently preferred embodiment of the invention is set forth in Part I. It should be noted that reference numbers appearing in the drawings and in the following descriptions are comprised of three digits. The two least significant (rightmost) digits identify a particular element appearing in a particular drawing and the most significant (leftmost) digit identifies the figure in which that element first appears. For example, element 124 is the 24th element appearing in FIG. 1 and first appears in FIG. 1. A reference numbers is assigned the first time the reference element appears in the descriptions and is used refer to that element throughout the following descriptions and drawings.

A. Elements of System 102 and General Operation (FIG. 1).

Referring to FIG. 1, therein is presented a block diagram of an exemplary System 102 incorporating the inter-processor bus structure of the present invention. As shown therein, the two primary elements of the system bus structure that are visible at this level are System Bus 104 and System Bus Priority (SBP) Bus 106. System Bus 104, the detailed structure of which is described further below, is the means by which the elements of System 102 communicate with one another. SBP Bus 106, also described in detail below, is the link through which the elements connected from System Bus 104 determine access to System Bus 104.

As shown in FIG. 1, the elements comprising System 102 fall into two classes, those which are connected directly to System Bus 104 and SBP Bus 106 and those which are connected indirectly to System Bus 104, that is, through another element which in turn is connected directly to System Bus 104 and SBP Bus 106. As described further below, elements directly connected to System Bus 104 and SBP Bus 106 are fundamentally regarded as a peers with regard to access to System Bus 104, that is, each such element has equal priority of access to System Bus 104 with respect to all other such elements. The access priorities of indirectly connected elements are determined by the access priorities of the associated elements through which they are connected to System Bus 104. Considering first the elements connected directly to System Bus 104, each such element includes sufficient internal intelligence, for example, in the form of microcode control, to perform at least specialized functions independently of the other elements of System 102. Examples of such elements, as illustrated in FIG. 1, includes Memory Units (MEMs) (1 to n+2) 108, Central Processing Units (CPUs) (1 to n+1) 110, Local System Controllers (LSC) 112, Remote System Controllers (RSC) 114, and System Bus Interfaces (SBIs) (1 to n+1) 116.

The design of and functions performed by elements such as MEMs 108 and CPUs 110 are well known in the art and require no further description. LSC 112 and RSC 114 may, for example, be small computers of the personal or professional class adapted to perform certain system control functions, such as providing a user control interface, that is, a "soft control panel". In this respect, RSC 114 may differ from LSC 112 in being connected to a remote user/controller, for example, for diagnostic purposes, through a Telecommunication Link (TC).

SBIs 116 are the elements, described above, through which indirectly connected elements such as Satellite Processing Units (SPUs) 118 are provided with access to System Bus 104. As illustrated in FIG. 1, SPUs 118 are interconnected with each other and to an associated SBI 116 through an SPU Bus 120. The link between the associated SPU Bus 120, and thus the SPUs 118 connected therefrom, and System Bus 104 is in turn provided through the associated SBI 116. It should be noted that SPUs 118 and SPU Buses 120 may use the same structure and operation as described below with reference to System Bus 104 and SBP Bus 106, that is, may be arranged as local system buses identical to the main system bus. SPUs 118 and SPU Buses 120 may alternately be of any other suitable design for the system operation.

SPUs 118 essentially include all devices or system elements which, for example, due to data rates or functions, do not require direct access to System Bus 104 to perform their functions. Examples of SPUs 118 are input/output devices such as disc drives, displays, printers, telecommunications links, tape streamers and user terminals. SPUs 118 may further includes independent or associated processing units, such as other general purpose computers or specialized processing devices,

such as scanners and specialized arithmetic or signal processors.

B. System Bus 104, General Structure and Operation (FIGS. 1, 1A and 1B)

Returning to System Bus 104, as described above System Bus 104 is the means through which the elements of System 102 communicate. In the present embodiment of System 102, and as shown in FIG. 1, System Bus 104 is a linear bus with each of the peer elements of System 102 connected therefrom, the connections to System Bus 104 being unidirectional or bidirectional as required by the function of the element. System Bus 104 may be extended as required by the particular configuration of a System 102, that is, to add or subtract system elements or to connect two or more System 102's into a single system.

It should be noted that, as described below, the logical configuration of System Bus 104 is defined by SBP Bus 106 and may assume any topological structure required by the function of System 102. For example, System Bus 104 may be physically arranged in the loop and star configurations illustrated respectively in FIGS. 1A and 1B. In the loop configuration, the ends of System Bus 104 are tied together to form a closed loop from which System Elements (SEs) 122 are connected. In the star configuration, System Bus 104 is comprised of a number of bus segments radiating from a common junction and SEs 122 are connected from the radiating segments as required by the system configuration.

C. System Bus Priority Bus 106 (FIG. 1)

Referring again to FIG. 1, as described in detail further below SBP Bus 106 is the means through which the System 102 elements connected to System Bus 104 determine access to System Bus 104. As shown in FIG. 1, SBP Bus 106 forms a loop with all of the elements connected from System Bus 104 being serially connected in the SBP Bus 106 loop.

It is assumed, in the exemplary System 102 presented herein, that all processing elements connected from System Bus 104 may have the capability to independently initiate interprocessor communications; thus all elements connected from System Bus 104 are shown as connected in the SBP Bus 106 loop. In certain cases, for example, memory elements, the processing elements may be such that they do not initiate interprocessor communications but will only receive and respond to such communications. Such elements will require access to System Bus 104 to receive such communications and to respond to such communications, for example, by reading data from a memory element to a CPU element, but will not be required to claim access to System Bus 104, that is, access to System Bus 104 will be provided by the element sending the communication being responded to. In such cases, these "response only" elements need not be connected in the SBP Bus 106 loop but will be connected to System Bus 104.

As described below, priority of access to System Bus 104 is passed from one element of System 102 to the next element in the SBP Bus 106 loop in a "rotating daisy chain". That is, if a given element currently has access to System Bus 104, the next element along the SBP Bus 106 loop following the current element has the highest priority for next access to System Bus 104, followed by the next element along SBP Bus 106, and so on around the SBP Bus 106 loop until the current element is reached again. When the element currently having access releases System Bus 104, the opportunity to gain next access is passed through SBP Bus 106 to the next element along SBP BUS 106. That next element may take access to System Bus 104 or, if it does not do so, passes the opportunity for access to its next element along SBP Bus 106, and so on until the element originally having access is reached again or some element along SBP Bus 106 takes access to System Bus 104. The order of priority of access to System Bus 104 thereby rotates around SBP Bus 106 with each element in turn having an opportunity to gain access to System Bus 104. Thus the average priorities of access to System Bus 104 of all elements connected thereto will be equal, with the relative priorities of the elements at particular points in time being determined by their positions along SBP Bus 106 relative to the element currently having either actual access to or the right to access System Bus 104.

Because of the rotating shifting of access priority to System Bus 104 among the elements of System 102 connected from SBP Bus 106, the elements connected to System Bus 104 do not contend for access to System Bus 104. As a result, the access determination logic, described further below, is simplified and, by eliminating System Bus 104 overhead which would otherwise be used in resolving access priorities, the speed of communication between the elements connected to System Bus 104 is increased. In addition, and because each element connected to System Bus 104 and SBP Bus 106 has an equal opportunity to gain access to System Bus 104, no element can be locked out of access to System Bus 104 for an extended period.

Moreover, and again because of the rotating shifting of access priority to System Bus 104 among the elements of System 102, the position of a System 102 element along either SBP Bus 106 or System Bus 104 has no bearing on the average priority of that element to access System Bus 104. That is, and as described above, all elements connected to System Bus 104 and in the SBP Bus 106 loop are peers having, on the average, equal access rights to System Bus 104. As such, an element may be added to System 102, or moved from one point along System Bus 104 and SBP Bus 106 to another, without effecting the average relative priorities of access to System Bus 104 of that element or any of the other elements connected to System Bus 104.

In this regard, SBP Bus 106 is represented in FIG. 1 as comprising a simple, clockwise loop with each element of System 102 being connected in series around the loop. It should be noted, however, that this representation is selected only for clarity of presentation. The elements of System 102 connected from System Bus 104 may, in fact, be connected in series along SBP Bus 106 in any desired order.

The order of elements along SBP Bus 106 may be effected, for example, by the above described temporary priorities of access having effect whenever a certain element has access to System Bus 104. That is, whenever a given element has access to System Bus 104, the next element along SBP Bus 106 has highest priority of next access, and so on around the SBP Bus 106 loop. If, for example, it were known that a particular operation involving access to System Bus 104 by a first element were frequently followed by a related operation again involving access to System Bus 104 by a second element, the second element may be connected in SBP Bus 106 next after the first element. Thus, whenever the first element executed its operation, it would be

known that the second element would have the highest priority of next accessing System Bus 104.

To illustrate with reference to FIG. 1, assume that SBI1 116 is primarily engaged in input/output operations of information being operated upon by CPU1 110, for example, text processing. In such a case, and in order to enhance speed of response to user operations, it may be desirable to allow CPU1 110 next access to System Bus 104 each time SBI1 116 transfers information from a user, that is, a SPU 118, to one of MEM1-n 108. In this case, then, CPU1 110 could be connected in the SBP Bus 106 loop next after SBI1 116.

Finally, a second element of SBP Bus 106 is illustrated in FIG. 1 and referred to as Local Priority Link (LPL) 124. LPL 124 is essentially a means by which the relative priorities of elements interconnected through LPL 124 may be fixed, as opposed to the rotating priorities determined by SBP Bus 106. As will be described in detail in a following description of the SBP Bus 106 element residing in each element connected therefrom, LPL 124 allows the fact of a pending requirement for access to System Bus 104 by one element to be passed to another element connected along a LPL 124 to inhibit any pending accesses to System Bus 104 in the second element.

D. System 102 Bus Structure (FIG. 2)

Referring to FIG. 2, therein is presented a diagrammic representation of System 102's bus structure. As described above and shown in FIG. 1, this structure includes System Bus 104, SBP Bus 106 and, in certain cases, an associated LPL 124.

D.1 Memory Control Bus 202: Memory Operations and Interprocessor Communications

As shown in FIG. 2, System Bus 104 includes a plurality of multiple and single line sub-busses. The first of these sub-busses is Memory Control (MC) Bus 202 which, upon the occurrence of a System 102 element obtaining access of System Bus 104, is used to communicate the type of System Bus 104 operation to be performed.

That is, when an element takes control of System Bus 104 that element signals this access by driving SBP Bus 106 to a state indicative of this fact and places on MC Bus 202 a code indicating the type of System Bus 104 operation to be performed. The elements of System 102 connected to System Bus 104 detect the occurrence of a System Bus 104 access by monitoring the state of SBP Bus 106 and, when an access is indicated, determine the type of System Bus 104 operation to be performed by reading the code placed on MC Bus 202 by the element having access to System Bus 104.

Most System Bus 104 operations are memory related, that is, are reads from or writes to MEM1 108. As such, and as will be seen below with reference to the MC Bus 202 codes, the entire class of non-memory related operations are indicated by a single code indicating that an "interprocessor" communication is to be executed, that is, a communication between two non-memory elements, such as an SBI 116 board and a CPU 110. As described below, the elements connected to System Bus 104 must in such cases refer to other of the System Bus 104 sub-busses to determine and execute interprocessor communications.

The MC Bus 202 codes provided in the present implementation of System 102 include:

CODE	TYPE OF OPERATION
0	No operation;
3	Read the contents of an MM 108 control register;
4	Read a quad word (16 bytes) of information from a specified MM 108 address location;
5	Read an octal word (32 bytes) of information from a specified MM 108 address location;
6	Read a double word (8 bytes) of information from a specified MM 108 address location;
7	Read a word (4 bytes) of information from a specified MM 108 address location;
8	Perform an inter-processor communication;
B	Write to an MM 108 control register;
C	Write a byte into a specified MM 108 address location;
D	Write a half word (2 bytes) into a specified MM 108 address location;
E	Write a double word into a specified MM 108 address location; and,
F	Write a word into a specified MM 108 address location.

It should be noted that the above codes are presented in hexidecimal form and that codes 1, 2, 9 and A are reserved for future use.

Interprocessor communications are thereby executed as a default case from memory related operations. That is, a short "decision branch", reference to a code on MC Bus 202, is provided to identify and initiate memory related operations while a longer "decision branch", reference to further information on other sub-busses of System Bus 104 is required for non-memory related operations. This method thereby effectively increases the speed with which the majority of System Bus 104 operations, that is, memory related operations, may be initiated and executed by providing a shorter decision path for such operations while retaining flexibility in defining and executing all types of System Bus 104 operations.

D.2 System Address (SA) Bus 204 and System Data (SD) Bus 206

The next major sub-busses of System Bus 104 are System Address (SA) Bus 204 and System Data (SD) Bus 206. Considering first memory related operations, SA Bus 204 is the means by which read and write addresses are communicated between elements requesting memory operations and the MEMs 108 executing the operations while SD Bus 206 is the means by which information is communicated between the MEMs 108 and the other elements of System 102.

D.2.a Memory Operations

In a memory operation, as described above the System 102 requesting a memory operation first gains access to System Bus 104 through the operation of SBP Bus 106, described in further detail below, and places an appropriate MC Bus 202 code on MC Bus 202 to indicate the type of operation to be performed. The requesting element then places the read or write address onto SA Bus 204 and, if the operation is a write, places the data to be written onto SD Bus 206. The addressed MEM 108 then writes the data into the corresponding storage location therein. If the operation is a read, the addressed MEM 108 reads the information from the addressed storage location and places the information on SD Bus 206, from which the information is read by the requesting element. In the present implementation

of System 102, for example, SA Bus is 24 bits wide, expandable to 31 bits, while SD Bus 206 is 64 bits, or a double word, wide.

Associated with SA Bus 204 and SD Bus 206 are three further single line sub-busses whose primary functions relate to memory operations. The first of these is WAIT 208. This signal is asserted by an addressed MEM 108 during a memory read operation if the requested information is not available and is monitored by the requesting element, which may accordingly go into a wait mode until the information becomes available.

The second memory operation control is BUSY 210, which is asserted by an addressed MEM 108 during a memory operation and before a System Bus 104 transmission is initiated. BUS 210 indicates that System Bus 104 is not available and is monitored by the elements of System 102.

The third memory operation control is Valid Memory Access (VMA) 212, which is asserted by an addressed MEM 108 to indicate that a requested memory operation is valid, that is, that the address or data are valid. VMA is monitored by the element requesting the memory operation to determine whether the request was successful, that is, valid.

D.2.b Interprocessor Communications (FIG. 2A)

Now considering non-memory related operations, that is, interprocessor communications, SA Bus 204 and SD Bus 206 operate differently in certain respects from that described above when an interprocessor operation is to be performed. As described above interprocessor operations are treated as a default from memory related operations. That is, a single MC Bus 202 code indicates the entire class of non-memory type operations. As also described above, upon the appearance of the interprocessor communication code on MC Bus 202 the elements connected to System Bus 104 must refer to information presented on SA Bus 204 and SD Bus 206 by the requesting element to determine the type of interprocessor operation to be executed.

Referring to FIG. 2A, therein is represented the information which may be presented upon SA Bus 204 and SD Bus 206 in an interprocessor operation. As shown therein, the information appearing on SA Bus 204 includes a 4 bit Target Address (TA) Field 214 identifying the target, or intended recipient of the message, a 4 bit Message Type (MT) Field 216 identifying the type of message to be sent to the target, and a 16 bit Message (ME) Field 218 which may contain a message. In certain interprocessor communication operations, wherein data is to be transmitted from one element to another, SD Bus 206 may contain a data field of up to 8 bytes.

D.2.b.1 TA Field 214 Codes

Considering now the various interprocessor communication fields appearing on SA Bus 204, the TA Field 214 may, for example, contain the following target identification codes:

CODE	TARGET IDENTIFIED
0	Support Control Unit (e.g., LSC 112 or RSC 114);
1	Broadcast to all CPUs 110;
2	CPU1 110;
3	CPU2 110;
4	CPU3 110;
5	CPU4 110;

-continued

CODE	TARGET IDENTIFIED
6	CPU5 110;
7	CPU6 110;
8	CPU7 110;
9	CPU8 110;
A	Reserved for future use;
B	SBI1 116;
C	SBI2 116;
D	SBI3 116;
E	SBI4 116; and
F	Broadcast to all SBIs 116.

It should be noted that the above codes are presented in hexidecimal format.

It is apparent from the above code formats that the exemplary system envisioned in the above code assignments includes a single Support Control Unit 112 or 114, up to 8 CPUs 110 and up to ~4 SBIs 116. The assignment of target codes may be altered at will, depending upon the envisioned configuration of the particular System 102.

It should be noted that SPUs 118 are targeted and messages transmitted thereto through the SPU 118's associated SBIs 116. It should also be noted that the interprocessor communications allow the simultaneous broadcast of messages to all elements of a given type, for example, to all CPUs 110 or to all SBIs 116.

There are no target identification codes for memory elements, that is, for MEMs 108, provided in the exemplary TA Field 214 codes. As described previously, all memory related operations are initiated at the MC Bus 202 code level and the target MEMs 108 identified by addresses concurrently appearing on SA Bus 204.

D.2.b.2 MT Field 216 Codes

The contents of the MT Fields 216 depend upon the particular type of recipient identified in the associated TA Field 214, that is, in the present example, whether the targeted recipient is an SBI 116, that is, an SPU 118 connected from an SBI 116, a CPU 110 or Support Control Unit 112 or 114. As will appear in the exemplary MT Field 216 codes presented below, an MT Field 216 code may identify a message as being the transfer of a message, the transfer of data, or a command for an operation or change of operating state on the part of the recipient element.

Considering first examples of the types of MT codes which may be transmitted to an SBI 116 type of element:

CODE	MESSAGE TYPE
0	Message transfer to target SPU 118;
1	Data transfer to target SPU 118;
8	Reset target SBI 116;
9	Reset target SPU 118;
A	Turn Input/Output (I/O) protection off;
B	Turn I/O protection on;
C	Enable I/O access to specified memory page; and
D	Disable I/O access to specified memory page.

Again, the MT Field 216 codes above are presented in hexidecimal format and codes, 2, 3, 4, 5, 6, 7, E and F are reserved for future use.

Considering now examples of the MT field 216 codes which may be used when the targeted recipient is a CPU 110:

CODE	MESSAGE TYPE
0	Class 1 I/O Interrupt;
1	Class 2 I/O Interrupt;
8	Interprocessor communications; and,
9	Synchronize clock.

Again, the codes are presented in hexidecimal format and codes 2 to 7 and A to F have been reserved for future use.

It should be noted that the above CPU 110 message types provide for two classes of I/O interrupt, Class 1 for when no error has appeared in the I/O operation and Class 2 for when an error has occurred in the I/O operation, for example, in the data. The two classes are provided because of the different handling of these events by the targeted CPU 110.

In the Interprocessor communication type of message, MT Code 8, only SA Bus 204 is used for the communication and the recipient CPU 110 refers to the accompanying ME Field 218 appearing on SA Bus 204 to determine the message, examples of which are presented below. The occurrence of such a communication causes an interrupt pending flag to be set in an interprocessor communication register internal to the recipient CPU 110, described below. The recipient CPU 110 micromachine will then read the interrupt pending flag, execute a macrointerrupt at the next opportunity, and execute a routine to appropriately handle the received message. If the interprocessor communication interrupt flag in the recipient CPU 110 is already active, the recipient CPU 110 will send to the transmitting CPU 110 acknowledgement and busy signals, as described below. It should be noted that in communications from SBIs 116 to CPU's 110, the SBI 116 receiving a busy response will handle the busy response in a manner appropriate to the SBI 116's bus protocol, that is, its protocol with respect to its SPUs 118

Again, the particular message type codes appearing in MT Field 216 and their meanings may be determined at will, depending upon the configuration and function of the System 102 envisioned and the elements appearing therein.

D.2.b.3 ME Field 218 Messages

The 2 byte interprocessor communicating messages appearing in ME Field 218 are, again, dependent upon the function and configuration of System 102 and the elements comprising System 102. Examples of such messages include, in the present implementation, communications between the system elements and the System 102's operating system being executed in the CPUs 110. Such operating system communications may include communications between an I/O device, that is, an SBI 116 or SPU 118, and the operating system and communications between the operating system and a Support Control Unit 112 or 114. For example, the operating system may send a message to a Support Control Unit to read or update an error file or requesting the Support Control Unit to examine some aspect of System 102's operation and report the findings of the examination to the operating system, for example, in diagnostic operations.

Other examples include communications between the CPUs 110 of a multiple-processor configuration of System 102. For example, a particular CPU 110 may wish exclusive access to a given page of memory and may inquire of the other CPUs 110 whether they are using that memory page. The requesting CPU 110 may then, if that page is not being used by another CPU 110, inform all CPUs 110, by a broadcast communication, that it is claiming exclusive access to that memory page. In a further example, a CPU 110 may wish to open and modify a particular file and will inform all other CPUs 110 that it is obtaining exclusive access to that file for that purpose. Yet other examples are messages coordinating the activities of the CPUs 110; for example, a first CPU 110 may assign a task to a second CPU 110 by an interprocessor message and the second CPU 110 may send a communication informing the first CPU 110 when the task is completed.

Still other examples of interprocessor messages occupying the ME Field 218 are I/O messages, essentially commands from the CPUs 110 to the SBIs 116 or SPUs 118 to initiate or control the operations of these elements.

Finally, and referring again to FIG. 2, as described above with reference to memory related operations certain single line sub-busses of System Bus 104 are associated with interprocessor communication operations. Among these are Acknowledge (ACK) 220 and Target Busy (TB) 222. ACK 220 is asserted by the target element of an interprocessor communication when that target exists and acknowledges that the sending element is attempting to send an interprocessor communication to that target element. The sending element monitors ACK 220 to determine whether the attempt to send an interprocessor communication was successful.

TB 222 is asserted by the target element of an interprocessor communication to indicate that the target element is busy and cannot accept the interprocessor communication. The sending element monitors TB 222 and, if the TB 222 is asserted by the target element, will handle the condition depending upon the nature and function of the sending element.

Also associated with both interprocessor communications and memory related operations is LOCK 224. LOCK 224 may be asserted by the initiator of a memory related operation or interprocessor communication to lock out all other users of System Bus 104. LOCK 224 may be asserted, for example, when an element wishes to communicate a series of interprocessor communications or a series of memory operations. LOCK 224 is monitored by all elements connected to System Bus 104 and no user will attempt to obtain access to System Bus 104 while another element is asserting LOCK 224.

Finally, as indicated in FIG. 2, System Bus 104 may include a System Clock (SYSCLK) 226, which is provided to all users of System Bus 104, thereby achieving common timing for all such elements.

Having described the operation of the bus structure of System 102, the bus interface logic residing in each of the elements connected to System Bus 104, and their operation, will be described next below.

E. System Bus Interfaces (FIGS. 3A and 3B)

Referring to FIGS. 3A and 3B, therein are presented diagrammic representations of the interface circuitry provided in each element connected to System Bus 104 to interface these elements to System Bus 104. It should be noted that the logic and circuitry presented herein are illustrative and representative only and may be replaced by any logic or circuitry performing equivalent functions. In addition, where the design and operation

of the logic and circuitry presented herein will be well understood by one of ordinary skill in the art, that logic or circuitry will not be described in detail. The operation of the element interfaces will, however, be described in detail where relevant to the present invention.

E.1. SBP Bus 106 Interface (FIG. 3A)

Referring first to FIG. 3A, therein is presented the interface circuitry primarily concerned with requesting and gaining access to System Bus 104. As previously described, each element connected to System Bus 104 is connected in series along the SBP Bus 106 loop. Other sub-busses of System Bus 104 concerned in obtaining access to System Bus 104 are, as indicated in FIG. 3A, Busy 210 and LOCK 224 and, in certain cases, LPL 124.

As previously described, when an element currently having access to System Bus 104 releases System Bus 104, the opportunity to gain next access to passed through SBP Bus 106 to the next element along SBP Bus 106. That next element may take access to System Bus 104 or, if it does not do so, passes the opportunity for access to its next element along SBP Bus 106, and so on until the element originally having access is reached again or some element along SBP Bus 106 takes access to System Bus 104.

As shown in FIG. 3A, the input of SBP Bus 106 from the previous element along the SBP Bus 106 loop is designated as SBP Input (SBPI) and is connected to a first input of Request Gate (RG) 302. It should be noted that SBPI is shown as an active low signal, as are all other overlined signals. The output of RG 302 is connected to the continuation of the SBP Bus 106 loop to the next element along the SBP Bus 106 loop and is designated as SBP Output (SBPO).

A second input of RG 302 is connected from the Request (REQ) output of Request Enable Gate (REG) 304. REQ will be generated, as described below, when the associated element wishes to request access to System Bus 104 and other conditions, determined by the various inputs to REG 304, also described below, do not prevent the element from requesting access. REQ operates as an enabling signal with respect to RG 302 and SBPI, that is, SBPI will be passed through RG 320 as SBPO and thus the SBPI of the next element along the SBP Bus 106 loop if REQ is not asserted. If, of course, REQ is asserted, SBPI is prevented from passing through RG 302 to become SBPO and the next element along the SBP Bus 106 loop will not receive an SBPI, the present element having claimed access to System Bus 104. RG 302 thereby operates as a gate to prevent the propagation of the opportunity to obtain access to System Bus 104, that is, the element may "capture" the opportunity, and as a signal level restorer for the SBPI/SBPO signal propagated along the SBP Bus 106 loop.

It should be noted that due to the structure of the SBP Bus 106 interface logic, the SBP Bus 106 access signal will propagate around the SBP Bus 106 loop at a speed determined by the propagation time around the physical loop and the delays through the RGs 302 of the elements connected therefrom. As such, delays in gaining System Bus 104 access due to the operation of SBP Bus 106 and the element interfaces thereto are minimized and the overall speed of operation of System 102's bus structure enhanced.

Referring now to REG 304, as shown in FIG. 3A REG 304 receives a Bus Request (BREQ) signal from the element's internal control circuitry, described further below, when the element wishes access to System Bus 104. BREQ is gated, in REG 304 by inputs BUSY and LOCK, previously described, from respectively BUSY 210 and LOCK 224. In certain cases, also previously described, REG 304 is provided with a Local Priority signal from a previous element through an LPL 124. The enabling of REQ, and thus the capture of SBPI/SBPO thereby requires that BUSY, LOCK and a Local Priority signal, if any, not be asserted.

E.2 System Bus 104 Interface (FIG. 3B)

Referring to FIG. 3B, therein is presented a diagrammic representation of the general interface of System Bus 104 of an element connected to System Bus 104. As shown therein, the interface logic includes an Interprocessor Communication Control (IPCC) 306 having inputs connected from the sub-busses of System Bus 104 as indicated. Included among these inputs are the 4 bits of MC Bus 202, WAIT, BUSY, VMA, ACK AND TB, the functions of which have been described above. Also included in IPCC 306's inputs are the TA Field 214 from SA Bus 204.

These inputs essentially define the occurrence, type and state of execution of interprocessor communications and IPCC 306 is essentially comprised of decoding logic for decoding these inputs and providing corresponding outputs to the internal control circuitry of the element, for example, microcode control circuitry.

The design of such internal control circuitry and of such decoding logic as IPCC 306 is well understood by those of ordinary skill in the art, especially after the previous and following descriptions of the operation of System 102's bus structure.

Among the outputs provided by IPCC 306 are an Interprocessor Interrupt (IPCI) to the elements interrupt handler, as previously described, and ACK and TB signals to ACK Bus 220 and TB Bus 222.

IPCC 306 also provides an output indicating the occurrence of an interprocessor communication (IPC) to Interprocessor Communication Register (IPCR) 308 which, as shown in FIG. 3B, is a register connected from the MT Field 216 and ME Field 218 portions of SA Bus 204. IPCR 308 thereby captures and stores the MT and ME fields of an interprocessor communication appearing on SA BUs 204, and provides these fields as inputs to the elements Internal Control Logic (Internal Control) 310.

As is well known in the art, Internal Control 310 may be comprised, for example, of dedicated control logic or a microcode programmed microprocessor controller. Internal Control 310 operates in a first respect to control the operation of the element in response to a received interprocessor communication and, in a second respect, to initiate and control interprocessor communications from the present element.

In this second respect, Internal Controller 310 may generate interprocessor communication control signals to Interprocessor Communication Output Controller (IPCO) 312. IPCO 312 may in turn be comprised of registers and decoding logic to generate MT and ME fields to the SA Bus 204 when the element is initiating an interprocessor communication, and may generate the BREQ signal, described above, when the element wishes to request access to System Bus 104 for an interprocessor communication.

Finally, the interface circuitry may include Data Registers and Drivers (DR) 314 connected with SD Bus

206 to communicate data between the element and SD Bus 206. Such data communication may occur, as previously described, in a memory related operation or when data is transferred in an interprocessor communication.

The interface between an element and System Bus 104 may differ from that described above when that element is a memory element such as an MEM 108. In such a case, the circuitry illustrated in FIG. 3A may be modified accordingly, that is, may contain only those functions necessary for memory related operations.

For example, an MEM 108 IPCC 306 may be provided only with inputs from MC Bus 202, which are sufficient and complete to define all memory related operations. The MEM 108's IPCC 306 may, accordingly, provide outputs WAIT, BUSY and VMA to the appropriate System Bus 104 sub-busses, rather than those shown in FIG. 3A. Similarly, the IPCR 308 of an MEM 108 will be comprised of an address input register connected from SA Bus 204 for receiving memory read and write addresses, and the MM108 will include a bidirectional data connection to SD Bus 206.

Having described the bus and interface structures of System 102's bus structure, and the functions and operation of the signals involved therein, the operation of System 102's bus structure will be described further below with reference to flow charts illustrating these operations.

F. Flow Chart Illustration of Operation (FIGS. 4A and 4B)

Referring to FIGS. 4A and 4B, therein are presented flow chart illustrations of the operations of System 102's bus structure. FIG. 4A is an illustration of a system bus operation from the viewpoint of the bus requestor, while FIG. 4B is an illustration from the viewpoint of the target element.

F.1 Bus Requestor Operation (FIG. 4A)

Referring to FIG. 4A, an interprocessor bus operation, either a memory operation or an interprocessor communication, is begun with the requesting element generating, through its internal control logic, an Access Request. In the first step, the requestor determines whether System Bus 104 is available; that is, and as previously described, the requestor determines whether BUSY, LOCK and LPL 124, if any, are asserted and whether an SBP1 is available.

In the second step, and if the conditions for bus availability are met, the requestor seizes System Bus 104 by capturing the SBP Bus 106 signal and places an appropriate code on MC Bus 202 to indicate the type of operation to be performed. If System Bus 104 is not available, the requestor takes what action is necessary to wait until the bus is available and retries access.

If access to System Bus 104 was gained, and the operation is a memory related operation, the requestor places the memory operation address on SA Bus 204 and the information to be written, if any on SD Bus 206 and monitors WAIT Bus 208.

If the responding memory element is busy, the memory will assert WAIT on WAIT Bus 208 and the requesting element must enter a wait mode until the responding memory element is available.

If the responding memory element is available, and the memory request is valid, the memory element will assert VMA on VMA Bus 212 and the operation will be completed. If the memory request was not valid, VMA will not be asserted and the requestor must refer to its error handler facility to resolve the problem.

If the requested operation was an interprocessor communication, the requesting element will place the appropriate fields, as previously described, on SA Bus 204 and SD Bus 206 and will monitor ACK Bus 220 and TB Bus 222.

If the target element asserts ACK and does not assert TB, the operation will be completed.

If the target element asserts ACK and TB, the requestor must wait and retry the communication as the target element is busy.

If the target element does not assert and does assert TB then the target element is again identified as busy and the requestor must retry the communication. If the target element does not assert ACK or TB, then the requestor must refer to an error handler to resolve the problem.

F.2 Target Element Operation (FIG. 4B)

Referring now to FIG. 4B, therein is presented a general flow chart of the operation of a target element for both memory related operations and interprocessor communication operations.

At the first step, the target element monitors SBP Bus 106 and identifies the initiation of a System Bus 104 operation by the state of SBP Bus 106. At this time, the element reads the code on MC Bus 202 to determine the type of System Bus 104 operation to be executed.

Considering first a memory operation, and assuming that the target element is a memory element, the element then reads SA Bus 204 to determine the target address in System 102's address space and SD Bus 206 to capture the data to be written if the operation is a memory write.

If the target element is busy, it will respond by asserting BUSY on BUSY Bus 210. If the target element is not busy and the memory request is valid, the target element will assert VMA on VMA Bus 212 and will perform the requested operation.

Considering finally an interprocessor communication, the target element determines, by reading the MC Bus 202 code, that an interprocessor communication is to be performed and then reads the TA Field 214 from SA Bus 204 to determine whether that element is the target element.

If that element is the target element, it asserts ACK and, if not busy, does not assert BUSY.

Assuming that the element is not busy and is the target element, the element reads the MT Field 216 from SA Bus 204 to determine the type of communication and the ME Field 218 from SA Bus 204 to capture the message. The element then responds to these fields by executing what operations are required by the message, including reading data from the SD Bus 206 if the communication requires the transfer of data.

ADDITIONAL DESCRIPTION OF SPB BUS 106 AND RELATED PRIORITY LOGIC: FIGS. 6 and 7

As described in the foregoing and shown in FIG. 1, system priority bus (SPB) 106 connects the components of system 102 into a circular daisy chain. As further described in the foregoing, the relative priorities of the components of system 102 are not static, but are instead determined relative to a given component of system 102. During operation of system 102, the component of system 102 from which priority is determined rotates around the circular chain defined by SBP Bus 106. The

following additional description provides details of a presently-preferred embodiment of SPB Bus 106 and of the priority logic. The additional description begins with an overview of the priority system and then describes the manner in which the priority system is implemented in the preferred embodiment.

G. Overview of a Preferred Embodiment of the Priority System: FIG. 6

FIG. 6 presents an overview of a presently-preferred embodiment of the priority system employed in system 102. In FIG. 6, the components of system 102 are represented by nodes 603. Each node 603 contains the priority logic by means of which one or more of the components of system 102 gains access to system bus (SB) 104. The number of nodes 603, and the number of system 102 components represented by a given node 603 depend of course on the configuration of system 102. When a node 603 serves more than one component of system 102, the relative priorities among the components served by the node are determined by additional priority logic, for example a static daisy chain 501 connecting the components served by node 603. In such a situation, the priority determined by the additional priority logic determines which of the devices served by node 603 actually receives access to SB 104 when node 603 gains access to SB 104.

Each node 603 is connected to SB 104 and receives at least BUSY 210, LOCK 234, and signals from MC bus 202. Each node 603 is further connected to SBP 106 to form a circular chain, represented in FIG. 6 as including nodes 603(1) through 603(n). In a preferred embodiment, SBP 106 has two components: priority line (P) 607, from which each node 603 determines whether it may access SB 104, and anchor line (A) 605, which permits the node 603 which has the current highest priority and which consequently currently serves as the "anchor" from which the priorities of the other nodes 603 are determined to rotate sequentially around the circular chain. As may be seen from FIG. 6, each node 603 receives A 605 and P 607 from the node 603 immediately preceding it in the circular chain and provides A 605 and P 607 to the immediately following node 603. In the following discussion, A 605 and P 607 carry subscripts indicating the node which is their source. For example, A 605(n) is provided from node 604(n) to node 603(1).

FIG. 6 shows only the internal structure of node 603(1) in detail; however, all other nodes 603 have similar internal structures and operate in similar fashion. There are four main components of the internal structure of node 603(1): priority logic (PL) 609, anchor logic (AL) 615, request logic (RL) 621, and node grant logic (NGL) 625. Beginning with PL 609, that logic receives as input from P 607(n) indicating whether any preceding node 603 has received the right to access the bus, an input 611 from AL 615 indicating whether node 603(1) is presently the "anchor" and therefore has the highest priority, and an input 613 from RL 621 indicating that one of the components represented by node 603(1) wishes to access the bus. PL 609 further produces output P 607(1), which indicates whether following nodes 603 may have the right to access SB 104 and output 617, which indicates whether node 603(1) is presently entitled to access SB 104.

If input 613 does not indicate a request, PL 609 simply passes the input it received from P 607(n) through to P 607(1). If input 613 does indicate a request and input P 607(n) indicates that node 603(1) may have the right to access SB 104, node 603(1) sets output P 607(1) to indicate that following nodes 603 may not have access and output 617 to indicate that node 603(1) is entitled to access SB 104. If input 613 does indicate a request, input P 607(n) indicates that a preceding node 603 has requested access, and input 611 indicates that node 603(1) is not the anchor, PL 609 again passes the value of P 607(n) through to P 607(1) is the anchor, PL 609 sets P 607(1) to indicate that following nodes 603 may not have access and output 617 to indicate that node 603(1) either when no preceding node 603 has requested access to the bus or when node 603(1) is the anchor, regardless of whether a preceding node 603 has requested access.

Continuing with AL 615, AL 615 includes a state element which indicates whether node 603(1) is presently the anchor. At any given time, that state element will indicate that a node 603 is the anchor in only one of the nodes 603 in the circular chain. AL 615 receives as inputs A 605(n), which indicates the value of the state element in the preceding node 603's AL 615 and signals received from SB 104 via BIN 627(1). AL 615 determines from these signals whether any component of system 102 has gained access to SB 104. AL 615 outputs the present value of the state element to PL 609 as described in the discussion of PL 609 and to A 605(1), which provides it to AL 615 of the following node 603. Each time AL 615 receives signals via BIN 627(1) indicating that a component has gained access to SB 104, AL 615 sets the state element to the value presently on A 605(n). AL 615 in all other nodes 603 works in the same fashion; consequently, each time a component gains access to SB 104, the value of the state element which indicates that a node 603 is the anchor moves to the following node 603. Thus, as components of system 102 use SB 104, the anchor node 603 rotates around the circular chain and each node 603 has the highest priority in turn.

Turning to request logic (RL) 621, RL 621 receives as an input node request (NR) signal 619, indicating that a component represented by node 603(1) is requesting access. As previously indicated, if node 603(1) represents several components, NR 619 may be produced by priority logic (not shown in FIG. 6) which determines which of the components represented by node 603(1) is to gain access to SB 104. RL 621's outputs are 613, which indicates to PL 609 that node 603(1) is making a request to access the bus, and 623, which indicates the same fact to NGL 625. NGL 625 determines whether the component which is requesting access will receive it. Inputs to NGL 625 in a preferred embodiment are BUSY 210 and LOCK 234 from SB 104, output 623 from RL 621 indicating that one of the components represented by node 603(1) is requesting access, and output 617 from PL 609, indicating that node 603(1) is requesting access and output 617 from PL 609, indicating that the node 603(1) is entitled to access. When BUSY 210 and LOCK 234 indicate that SB 104 is available, output 623 indicates that a request has been made, and output 617 indicates that node 603(1) is entitled to access SB 104, NGL 625 output node grant signal (NG) 629 to which the component represented by node 603(1) which has the right to access SB 104 responds by seizing the bus.

The effect of the operations just described is that when a component represented by node 603(1) requests access to SB 104, node 603(1) obtains access either if node 603(1) is the current anchor node or if neither the

current anchor node 603 nor any of the nodes 603 following the current anchor node 603 and preceding node 603(I) is requesting access. Further, each time any component of system 102 obtains access to SB 104, the node 603 following the current anchor node 603 becomes the new anchor node. The former anchor node 603 then has the lowest access priority; however, if the new anchor node 603 and none of the nodes 603 following the new anchor node and preceding the former anchor node requests the bus, former anchor node 603 may access it. Rotating circular chain 601 thus neither permits a single component of system 102 to "hog" SB 104 nor prevents any component from receiving immediate access to SB 104 when the priority of the component's node 603 permits it.

H. Detailed Description of Node Priority Logic: FIG. 7

FIG. 7 is a logic diagram showing an implementation of PL 609, AL 615, RL 621, and NGL 625 in nodes 603(1) and 603(n). The relationship between FIG. 6 and FIG. 7 is the following: PL 609 is implemented in the nodes of FIG. 7 by means of NOR gate 705 and NAND gate 709; AL 615 is implemented by means of anchor flip-flop (AFF) 701; RL 621 is implemented by means of node request flip-flop (NRFF) 703; and NGL 625 is implemented by means of AND gate 711 and NGFF 713. The discussion will first explain the logic of node 603(1) and then explain the differences between that node and the other nodes 603, exemplified by node 603(n).

Beginning with PL 609, in a preferred embodiment, P 607(n) is high when a preceding node 603 has requested access to the bus and low when no preceding node 603 has requested access. A 605(I), from AFF 701(1), which corresponds to input 611, is low when node 603(1) is the anchor node and otherwise high. NR 719(1), finally, which corresponds to input 613, finally, is high when a component represented by node 603(1) has requested access and otherwise low.

If node 603 is not requesting access, NR 719(1) is low and the value of P 607(1) is determined by the value of input 707(I), which in turn is determined by the value of P 607(n). Thus, when there is no request in node 603(1), the value output on P 607(1) is the same as that received on P 607(n).

If node 603(1) is requesting access, driving NR 719(1) high, and P 607(n) is low, indication that no preceding node has taken access, output 707(1) is low regardless of the value of A 605(1). When 707(1) is low and NR 719 is high, P 607(1) is high, indicating to nodes following node 603(1) that a preceding node has taken access. When P 607(n) is high, P 607(1) is also high, but the value of output 707(1) depends on the value of A 605(1). If it is high, indicating that node 603(1) is not the anchor node, 707(1) is high; if A 605(1) is low, output 707(1) is low. As will be described in more detail with regard to NGL 625, when 707(1) is low and other conditions are met, the component requesting the bus seizes the bus. Thus, the component can seize the bus in either of two cases: if no preceding node 603 has taken access of if a preceding node 603 has, but node 603(1) is currently the anchor node.

As should be apparent from the above discussion, which node 603 in rotating circular chain 601 is to obtain access to SB 604 cannot be determined until each node 603 beginning with the current anchor node has had a chance to respond to P 607. Thus, there must be a period for resolving access which is long enough to permit such response. The length of the period will depend on the gate delays in each node 603 and the number of nodes 603 in the circular chain. It is an advantage of the present implementation that gates 705(1) and 709(1) may be implemented by means of a single AND/OR invert gate. Consequently, only a single gate delay is necessary in each node 603. In system 102, the period for resolving access is 60 ns, and 6 ns are required for set up and gate delay in each node 603; thus system 103 may have up to 10 nodes 603.

Continuing with the implementation of AL 615, AFF 701(I) is a JK flip flop. The J input receives NOT A 605(n), while the K input receives A 605(n). In the AFFs 701 of nodes 603(2) through (n), the J input receives A 605 from the preceding node 603 and the K input receives that node's NOT A 605. When the AFFs 701 receive a rising edge on their clock inputs, the value on the J and K inputs are stored in the flip flops. In the preferred embodiment, the clock signal is F(BCOM) 714. This is a signal which is derived from signals on MC Bus 202 and bus clock signals on SB 104 and which indicates that a component of system 102 has seized the bus. The input of NOT A 605(n) to the J input of AFF 701(1) and of A 605(n) to the K input of that flip flop guarantees that exactly one AFF 701 will have a low Q output, indication that its node 603 is the anchor node, in circular chain 601, and the use of F(BCOM) 714 to set all of the AFFs 701 ensures that the anchor moves to he next node 603 each time a component of system 102 seizes SB 104. At initialization of system 102, an arbitrary one of the nodes 603 in the system is selected to be the first anchor node, and AFF 701 in that node is reset so that A 605 from that node is low. In FIG. 7, node 603(1) is the selected node, and NOT RES 721(1) permits resetting of AFF 701(1).

The implementations of RL 621 and NGL 625 are straightforward. RL 621 consists of NRFF 703, which receives and retains NR signal 619(I) indication that a component represented by node 603(I) has made a request. NOT RES 721(1) permits resetting NRFF 703(1) when the request has been services. NGL logic 625 consists of AND gate 711(1) and NGFF 713(1). AND gate 711(1) requires that NRFF 703(1) indicate a request, that neither BUSY nor LOCK is being asserted in system bus 104, and that output 707(1) indicates that node 603(1) may access the bus, either because no preceding node wants to access it or because node 603(1) is the current anchor node. If all of these conditions hold, NGFF flip flop 713(1) is set, NG 629(1) is high, and in response to that signal, the component which made the request seizes SB 104.

The foregoing discussions has disclosed how one skilled in the art may construct a dynamic circular priority chain in which the priority of a node with regard to access to a bus in a computer system is determined by its position on the circle relative to a current anchor node and in which the current anchor node changes each time a component attached to the bus accesses the bus. In such a dynamic circular priority chain, no node can "hog" the bus, and at the same time, any node can have immediate access to the bus if there is no node with higher priority requesting access. Moreover, the time to determine which node will receive access is no more than that required for a priority signal to propagate through priority logic implemented by means of single gate in each node.

I. Detailed Description of a Presently Preferred Embodiment of Node Priority Logic: FIG. 8

FIG. 8 is a logic diagram showing an implementation of PL 609, AL 615, RL 621, and NGL 625 in nodes 603(1) and 603(n). The relationship between FIG. 6 and FIG. 8 is the following: PL 609 is implemented in the nodes of FIG. 8 by means of NOR gate 705 and NAND gate 709; AL 615 is implemented by means of anchor flip-flop (AFF) 701; RL 621 is implemented by means of node request flip-flop (NRFF) 703; and NGL 625 is implemented by means of AND gate 711 and, in accordance with the invention, a programmable node grant counter NGCNT 720. The previous discussion as to the operation of the node logic shown in FIG. 7 applies in general also to the embodiment shown in FIG. 8 and will not be repeated. Instead only that portion of the logic that differs is discussed in detail.

The implementations of RL 621 and NGL 625 are straightforward. RL 621 consists of NRFF 703, which receives and retains NR signal 619(1) indicating that a component represented by node 603(1) has made a request. NOT RES 721(1) permits resetting NRFF 703(1) when the request has been serviced. NGL logic 625 includes AND gate 711(1), NGCNT 720(1) and inverter 724(1). AND gate 711(1) requires that NRFF 703(1) indicates a request, that neither BUSY nor LOCK is being asserted in system bus 104, and that output 707(1) indicates that node 603(1) may access the bus, either because no preceding node wants to access it or because node 603(1) is the current anchor node. If all of these conditions are met an enable count (ENA) input of NGCNT 720(1) is asserted and during a bus access cycle a transition of a clock signal (MEMCLK) causes NGCNT 720(1) to increment by one count.

Further in accordance with this aspect of the invention NGCNT 720(1) increments for each access to the bus, NGCNT 720(1) incrementing from some initially programmed value until a count of 15 (F_{16}) is reached. At this time a terminal count (TC) output of NGCNT 720(1) goes high, generating the signal NG 629(1). NG 629(1) being asserted causes an anchor out signal to be generated and applied to a next node in the priority chain. NG 629(1) is also inverted and fed back by inverter 724(1) and is applied to an active low load (LD) input of NGCNT 720(1). The low signal applied to the LD input causes NGCNT 720(1) to load the digital value appearing on lines 722(1) and to thereby preset NGCNT 720(1) to a predetermined value. Presetting NGCNT 720(1) to the predetermined value, other than 15, also causes the TC output to go low, removing the NG629(1) signal. Lines 722(1) may be sourced from manually set switches, hardwired jumpers or may be sourced from a latch or register (not shown) whereby the value may be programmed and changed, if desired, under a user's or an operating system's control. As can be appreciated, the value that is preset into the counter NGCNT 720(1) determines how many bus accesses the present highest priority anchor node is allowed before the highest priority is passed to a next requesting node in the chain of nodes. By example, if NGCNT 720(1) is preset to a value of seven then the node is granted eight consecutive bus accesses (15—7) before the highest priority is passed to the next requesting node. Of course, NGCNT 720(1) may be embodied in other than a four bit counter. For example, if NGCNT 720(1) were embodied in an eight bit counter up to 255 memory accesses could be made by the node before priority was rotated to another node. It should also be realized that the output of the counter could be compared to a digital value by a comparator circuit for determining when the programmed number of bus accesses have expired. That is, a number of possible hardware embodiments can be envisaged for realizing the desired result.

The foregoing discussion has disclosed how one skilled in the art may construct a dynamic circular priority chain in which the priority of a node with regard to access to a bus in a computer system is determined by its position on the circle relative to a current anchor node and in which the current anchor node changes after a predetermined number of accesses to the bus. In such a dynamic circular priority chain, no node can "hog" the bus, and at the same time, any node can have immediate access to the bus if there is no node with higher priority requesting access. Moreover, the time to determine which node will receive access is no more than that required for a priority signal to propagate through priority logic implemented by means of single gate in each node.

While the foregoing description has disclosed the best mode presently known to the inventor for implementing the invention, other implementations are possible, including ones in which no separate [illegible] is required for the anchor logic and ones in which an access to SB 104 by a node results in the next node following that node becoming the anchor node. Thus, the present embodiments are to be considered in all respects as illustrations and not restrictive, the scope of the invention being indicated by the appended claims rather that the foregoing description, and all changes which come within the meaning and range of equivalency of the claims are intended to be embraced therein.

What is claimed is:

1. Apparatus for determining priority of access to a bus among a set of devices coupled to the bus, each device being represented for priority purposes by a node in a group of nodes, each node being coupled to the bus and receiving a priority line from a first adjacent node and providing a priority line to a second adjacent node and having a priority relative to a single node with the highest priority, the priority determining apparatus comprising in each node:

priority logic means for permitting access to the represented device if no higher priority node has requested access; and

highest priority node specification means responsive to the bus for specifying whether the node is presently the highest priority node and, if the node is presently the highest priority node, dynamically giving the highest priority to another of the nodes in response to a predetermined number of accesses of the bus by one of the set of devices.

2. The apparatus set forth in claim 1 and wherein:

the node has the lowest priority after the highest priority node specification means gives the highest priority to another of the nodes.

3. The apparatus set forth in claim 1 and wherein:

the highest priority node specification means gives the highest priority to a node having the next highest priority.

4. Apparatus for determining priority of access to a bus among a set of devices coupled to the bus, each device being represented for priority purposes by a node in a group of nodes, each node being coupled to the bus and receiving a priority line from a first adjacent node and providing a priority line to a second adjacent

node and having a priority relative to a single node with the highest priority, the priority determining apparatus comprising in each node;

priority logic means for permitting access to the represented device if no higher priority node has requested access; and

highest priority node specification means responsive to the bus for specifying whether the node is presently the highest priority node and, if the node is presently the highest priority node, dynamically giving the highest priority to another of the nodes in response to a predetermined number of accesses of the bus by one of the set of devices, wherein

the highest priority node specification means includes counter means for counting a number of accesses to the bus and, responsive to a programmed value, gives the highest priority to another node on a next access by a node to the bus.

5. The apparatus set forth in claim 1 and wherein:

the group of nodes is configured as a circle for purposes of determining priority; and

the priority of a node is determined from the node's position in the circle relative to the highest priority node.

6. The apparatus as set forth in claim 5 and wherein:

the priority of the node is determined by the number of nodes between the node and the highest priority node in a preselected direction around the circle, with priority decreasing as the number of nodes increases.

7. The apparatus set forth in claim 6 and wherein:

the highest priority node specification means gives the highest priority to the adjacent node in the direction of decreasing priority.

8. The apparatus set forth in claim 5 and wherein:

the highest priority node specification means gives the highest priority to an adjacent node in the circle.

9. The apparatus set forth in claim 1 and wherein:

the group of nodes is configured as a circle for purposes of determining priority;

the priority logic means in each node includes preceding request value receiving means for receiving from the priority logic means of the immediately preceding node in a predetermined direction around the circle a request value which is in the alternative a first request value indicating that a preceding node beginning at the highest priority node has requested access to the bus and a second request value indicating that no preceding node has requested access,

preceding request value providing means for providing in the alternative the first or second request value to the priority logic means of the immediately succeeding node, and

priority value receiving means for receiving from the highest priority node specification means a priority value which is in the alternative a first priority value indicating that the node has the highest priority and a second priority value indicating that the node does not have the highest priority; and

the priority logic means operates responsively to the received request value, the received priority value, and any bus access request to respond to the absence of a bus access request and to any received request value by providing a request value equal to the received request value; respond to the presence of a bus access request and a received second request value by providing the first request value and permitting access to the bus,

respond to the presence of a bus access request and a received first request value and the second priority value by providing the first request value without permitting access to the bus, and

respond to the presence of a bus access request, a received first request value and the first priority value by providing the first request value and permitting access to the bus.

10. The apparatus set forth in claim 9 and wherein:

the highest priority node specification means includes preceding priority value receiving means for receiving the priority value specified by the immediately preceding node's highest priority node specification means,

preceding priority value providing means for providing the priority value specified by the node's highest priority node specification means to the immediately following node, and

bus access signal receiving means connected to the bus for determining when an access of the bus by one of the nodes has occurred; and

the highest priority node specification means sets the priority value to the priority value received on the preceding priority value receiving means when the bus access signal receiving means determines that an access of the bus has occurred.

11. Apparatus for determining priority of access to a bus among a set of devices coupled to the bus comprising:

a group of nodes, each node representing a device and having a priority and being responsive to requests to access the bus from the represented device to provide access if no device represented by a node with higher priority has requested access;

means for giving a single one of the nodes the highest priority;

means for giving the highest priority to a different node in response to a predetermined number of accesses of the bus by a device; and

means for determining the priority of each node relative to the highest priority node.

12. The apparatus set forth in claim 11 and wherein:

the means for determining the priority of each node relative to the node currently having the highest priority is a circular configuration of the nodes wherein the priority of a given node is determined by the node's position relative to the highest priority node.

13. The apparatus set forth in claim 12 and wherein:

the means for giving the highest priority to a different node gives the highest priority to an adjacent node in the circle.

14. The apparatus set forth in claim 13 and wherein:

the priority of a given node in the circular configuration is determined by the number of nodes between the node and the highest priority node in a preselected direction around the circle, with priority decreasing as the number of nodes increases; and

the means for giving the highest priority to a different node gives the highest priority to the adjacent node in the direction of decreasing priority.

15. A method for determining priority of access to a bus among a set of devices coupled to the bus, each device being represented for priority purposes by a node in a group of nodes and each node having a prior-

ity relative to a single node currently having the highest priority, the method comprising the steps of:

receiving an access request in a node from a represented device;

determining whether any node with a higher priority has received an access request;

if no such node has received an access request, permitting the device to access the bus;

counting a number of accesses by the device to the bus; and

in response to predetermined number of accesses to the bus, giving another node the highest priority.

16. A method as set forth in claim 15 and including an initial step of providing a value indicating a predetermined number of accesses that the device may make to the bus.

17. A method as set forth in claim 16 wherein the step of providing is accomplished by loading a digital value into a presettable counter means.

18. A method as set forth in claim 17 wherein the step of counting includes a step of incrementing the presettable counter means for each access by the device to the bus.

19. A method as set forth in claim 18 wherein the step of giving another node the highest priority occurs when the presettable counter means generates a terminal count output.

* * * * *

US005379379A

United States Patent [19]

Becker et al.

[11] Patent Number: 5,379,379

[45] Date of Patent: Jan. 3, 1995

[54] **MEMORY CONTROL UNIT WITH SELECTIVE EXECUTION OF QUEUED READ AND WRITE REQUESTS**

[75] Inventors: **Robert D. Becker, Shirley; Martin J. Schwartz, Worcester; Kevin H. Cureton, Pepperell; Kenneth J. Eng, West Bridgewater, all of Mass.**

[73] Assignee: **Wang Laboratories, Inc., Lowell, Mass.**

[21] Appl. No.: **580,365**

[22] Filed: **Sep. 6, 1990**

Related U.S. Application Data

[63] Continuation of Ser. No. 213,395, Jun. 30, 1988, abandoned.

[51] Int. Cl.[5] **G06F 13/16**

[52] U.S. Cl. **395/250**; 395/425; 395/725; 364/239.5; 364/242.91; 364/246.6; 364/DIG. 1; 364/937.01; 364/939.1; 364/969; 364/DIG. 2

[58] Field of Search ..... 364/DIG. 1, DIG. 2; 395/250, 425, 650, 725

[56] **References Cited**

U.S. PATENT DOCUMENTS

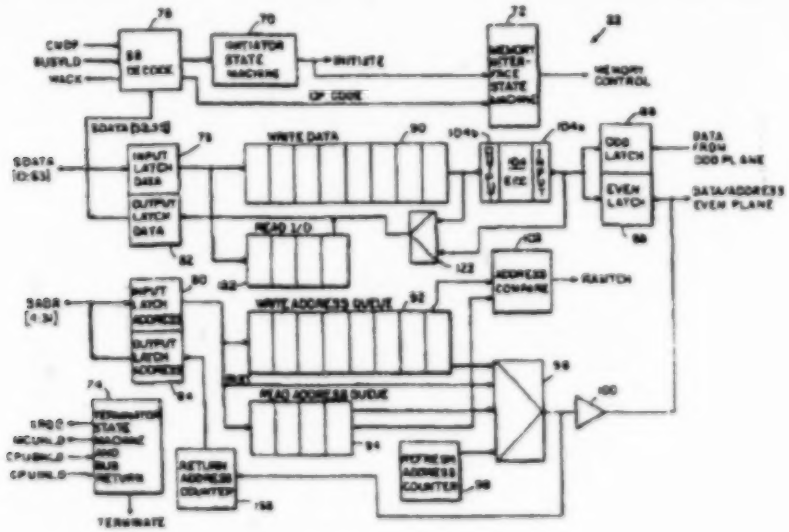
4,225,922	9/1980	Porter	364/200
4,354,232	10/1982	Ryan	364/200
4,516,199	5/1985	Frieder et al.	364/200
4,688,188	8/1987	Washington	364/900
4,783,736	11/1988	Ziegler et al.	395/425
4,792,926	12/1988	Roberts	365/189
4,833,651	5/1989	Seltzer et al.	365/189.07
4,864,543	9/1989	Ward et al.	365/221
4,933,901	6/1990	Tai et al.	365/189.07
5,034,922	7/1991	Burgess	365/189.07
5,097,442	3/1992	Ward et al.	365/78

Primary Examiner—Thomas G. Black
Assistant Examiner—Maria N. Von Buhr
Attorney, Agent, or Firm—Kenneth L. Milik

[57] **ABSTRACT**

A memory control unit (MCU) 22 includes a first interface for interfacing the memory control unit to one or more memory units; a second interface for interfacing the memory control unit to a system bus, including a system data bus for expressing information units, including memory read and write requests, and a system address bus. The MCU further includes logic, responsive to a write request from the system bus, for storing one or more information units within a memory unit at an address specified by the system address bus. The storing logic includes write request receiving and buffer logic having a plurality of storage locations for storing received write requests and associated write addresses prior to the execution of the write requests. The MCU further includes logic, responsive to a read request from the system bus, for reading one or more information units from a memory unit at a location specified by the system address bus. The reading logic includes read request receiving and buffer logic having a plurality of storage locations for storing received read requests and associated read addresses prior to the execution of the read requests. The memory control unit further has logic for comparing a received read address to write addresses stored in the write address buffer, the comparing logic having an output for indicating, when asserted, the occurence of the reception of a read address having a value within a predetermined range of values of one of the stored write addresses.

26 Claims, 10 Drawing Sheets



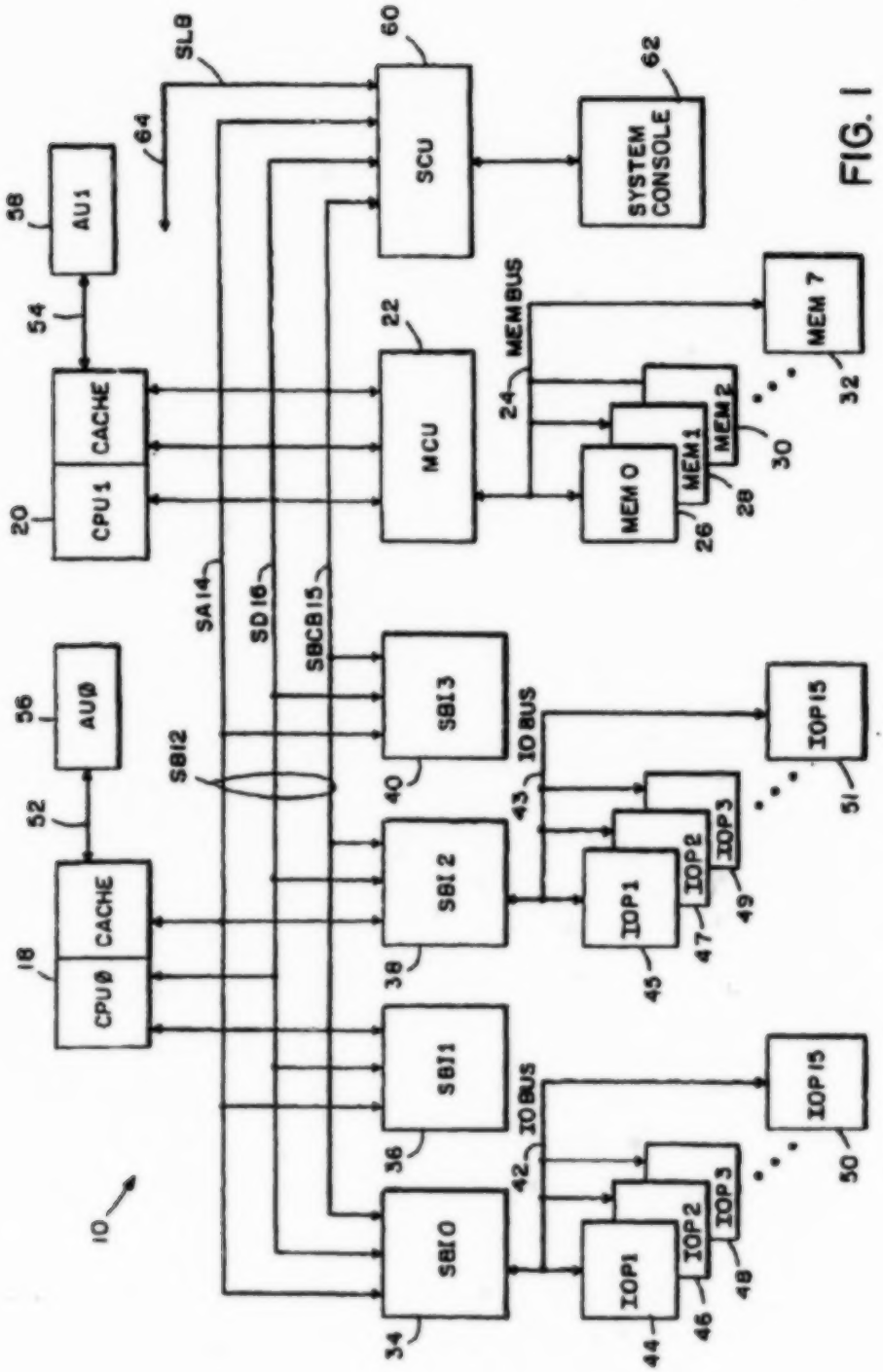

SL8
60
SCU
62
SYSTEM CONSOLE
64
38
AU1
54
CACHE
CPU1
20
22
MCU
MEM BUS
24
MEM 7
32
MEM 0
MEM 1
MEM 2
26
28
30
SA14
SD16
SBCB15
56
AUØ
52
CACHE
CPUØ
18
SBI2
SBI3
40
IO BUS
43
IOP15
51
IOP1
IOP2
IOP3
45
47
49
SBI2
38
SBI1
36
IO BUS
42
IOP15
50
IOP1
IOP2
IOP3
44
46
48
SBI0
34
10

FIG. 1

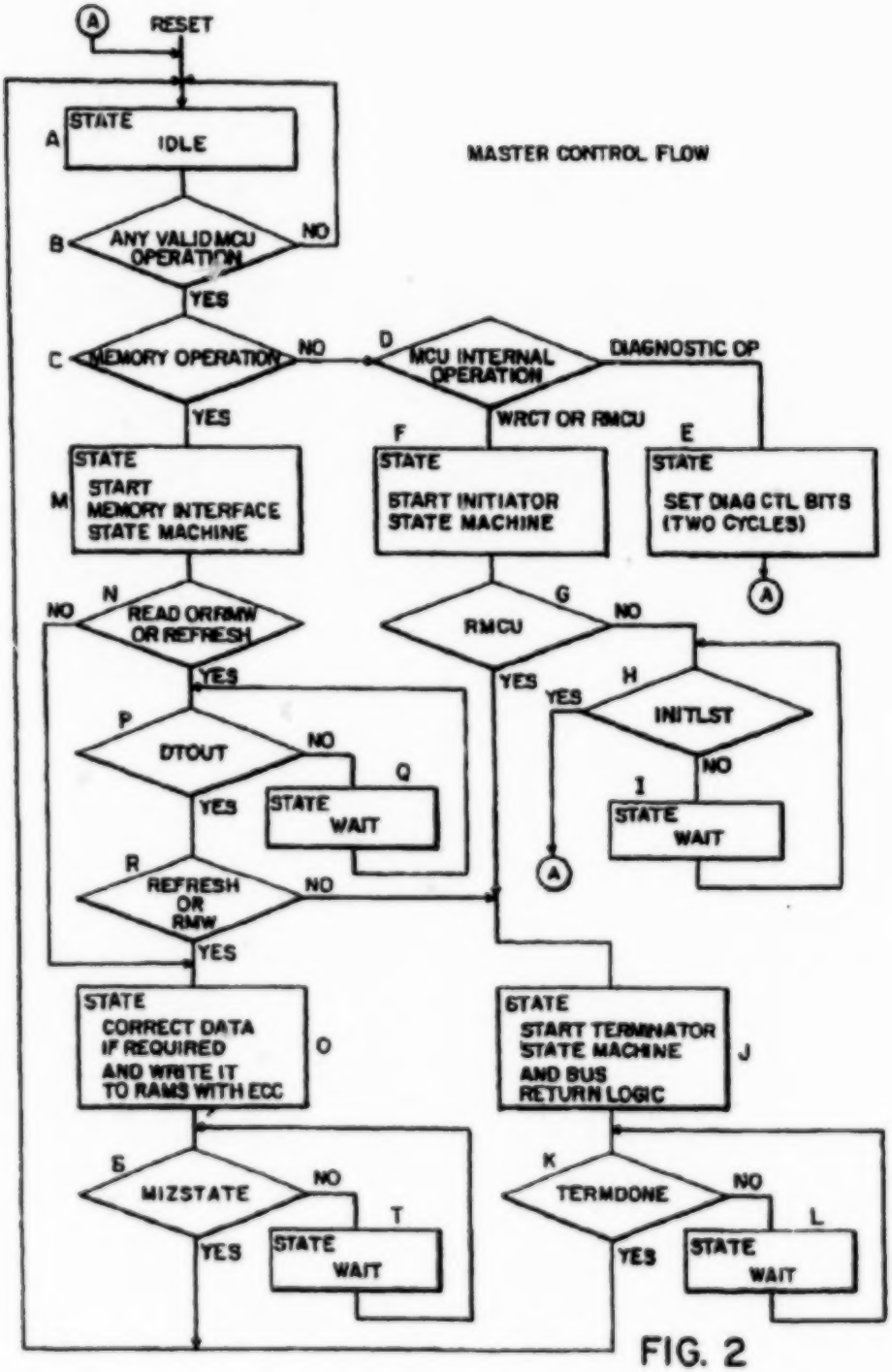

A
RESET
STATE
IDLE
MASTER CONTROL FLOW
B
ANY VALID MCU OPERATION
NO
YES
C
MEMORY OPERATION
NO
D
MCU INTERNAL OPERATION
DIAGNOSTIC OP
YES
WRCT OR RMCU
M
STATE
START MEMORY INTERFACE STATE MACHINE
F
STATE
START INITIATOR STATE MACHINE
E
STATE
SET DIAG CTL BITS (TWO CYCLES)
A
N
NO
READ OR RMW OR REFRESH
YES
G
RMCU
NO
YES
H
YES
INITLST
NO
P
DTOUT
NO
Q
STATE
WAIT
YES
I
STATE
WAIT
A
R
REFRESH OR RMW
NO
YES
STATE
CORRECT DATA IF REQUIRED AND WRITE IT TO RAMS WITH ECC
O
STATE
START TERMINATOR STATE MACHINE AND BUS RETURN LOGIC
J
S
MIZSTATE
NO
T
STATE
WAIT
YES
K
TERMDONE
NO
L
STATE
WAIT
YES

FIG. 2

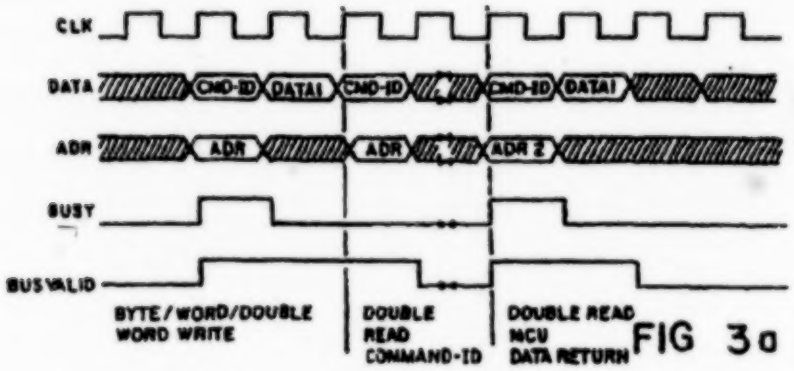


FIG 3a

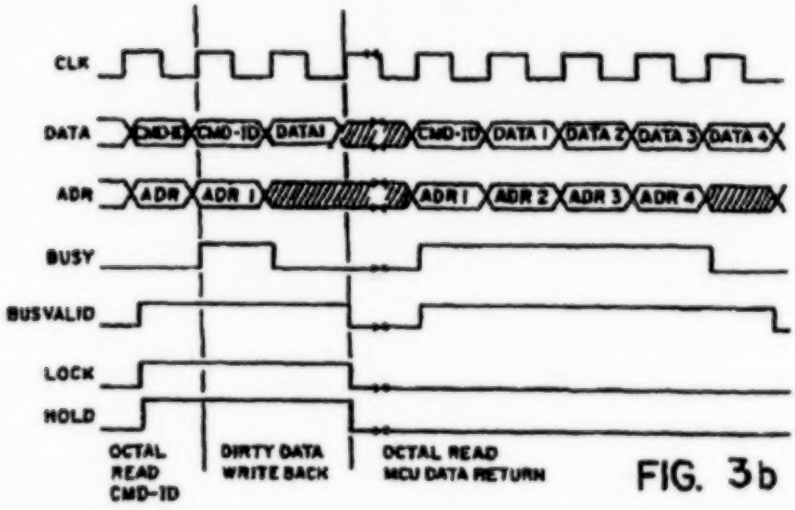


FIG. 3b

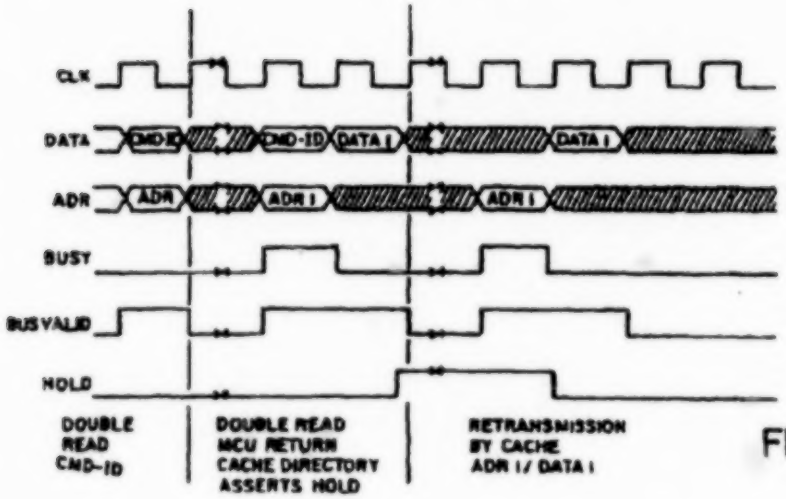


FIG. 3c

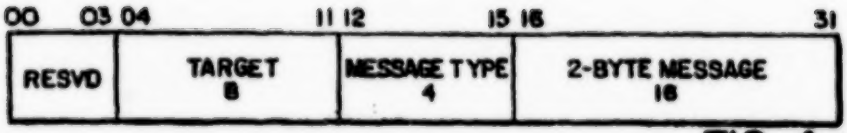


FIG. 4a

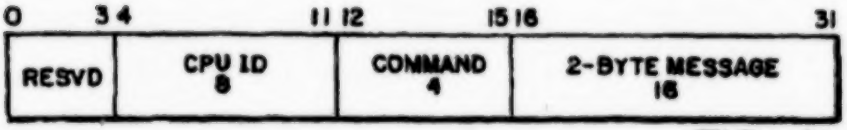


FIG. 4b

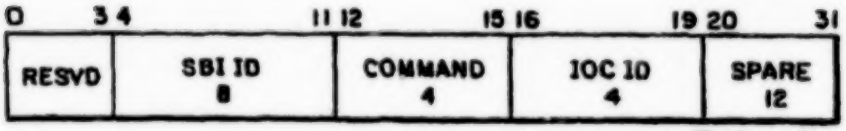


FIG. 4c

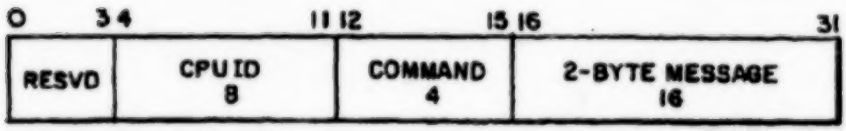


FIG. 4d

FIG. 4e

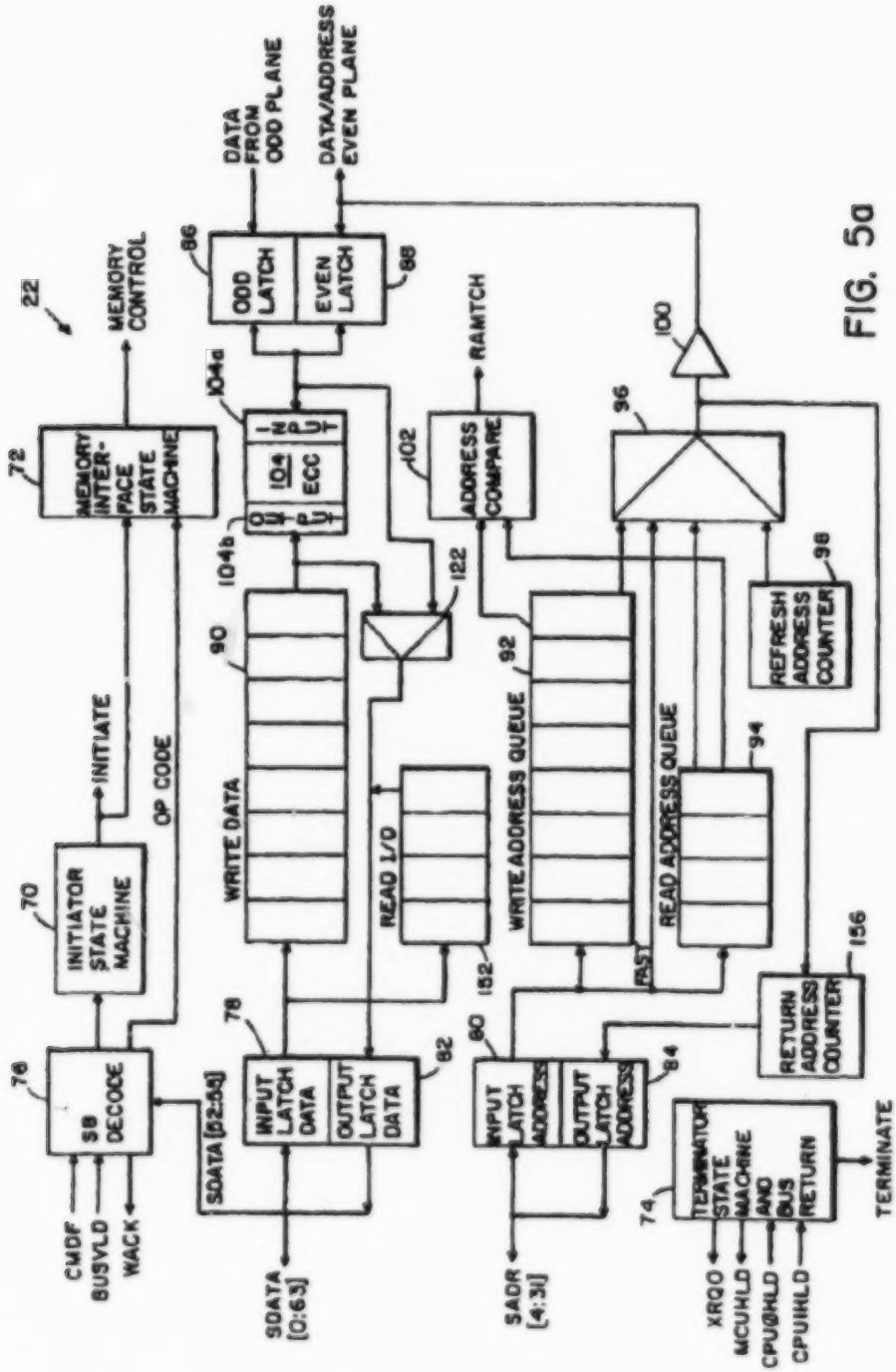

22
MEMORY CONTROL
DATA FROM ODD PLANE
DATA/ADDRESS EVEN PLANE
ODD LATCH
EVEN LATCH
86
88
104a
104b
I/O 104 ECC
RAMTCH
ADDRESS COMPARE
102
100
96
72
MEMORY INTERFACE STATE MACHINE
INITIATE
OP CODE
122
90
WRITE DATA
READ I/D
92
WRITE ADDRESS QUEUE
94
READ ADDRESS QUEUE
98
REFRESH ADDRESS COUNTER
70
INITIATOR STATE MACHINE
76
SB DECODE
78
INPUT LATCH DATA
OUTPUT LATCH DATA
82
SDATA [32:56]
152
FAST
80
INPUT LATCH ADDRESS
OUTPUT LATCH ADDRESS
84
156
RETURN ADDRESS COUNTER
74
TERMINATOR STATE MACHINE AND BUS RETURN
TERMINATE
CMDF
BUSYLD
WACK
SDATA [0:63]
SADR [4:31]
XRQO
MCUHLD
CPUØHLD
CPUIHLD

FIG. 5a

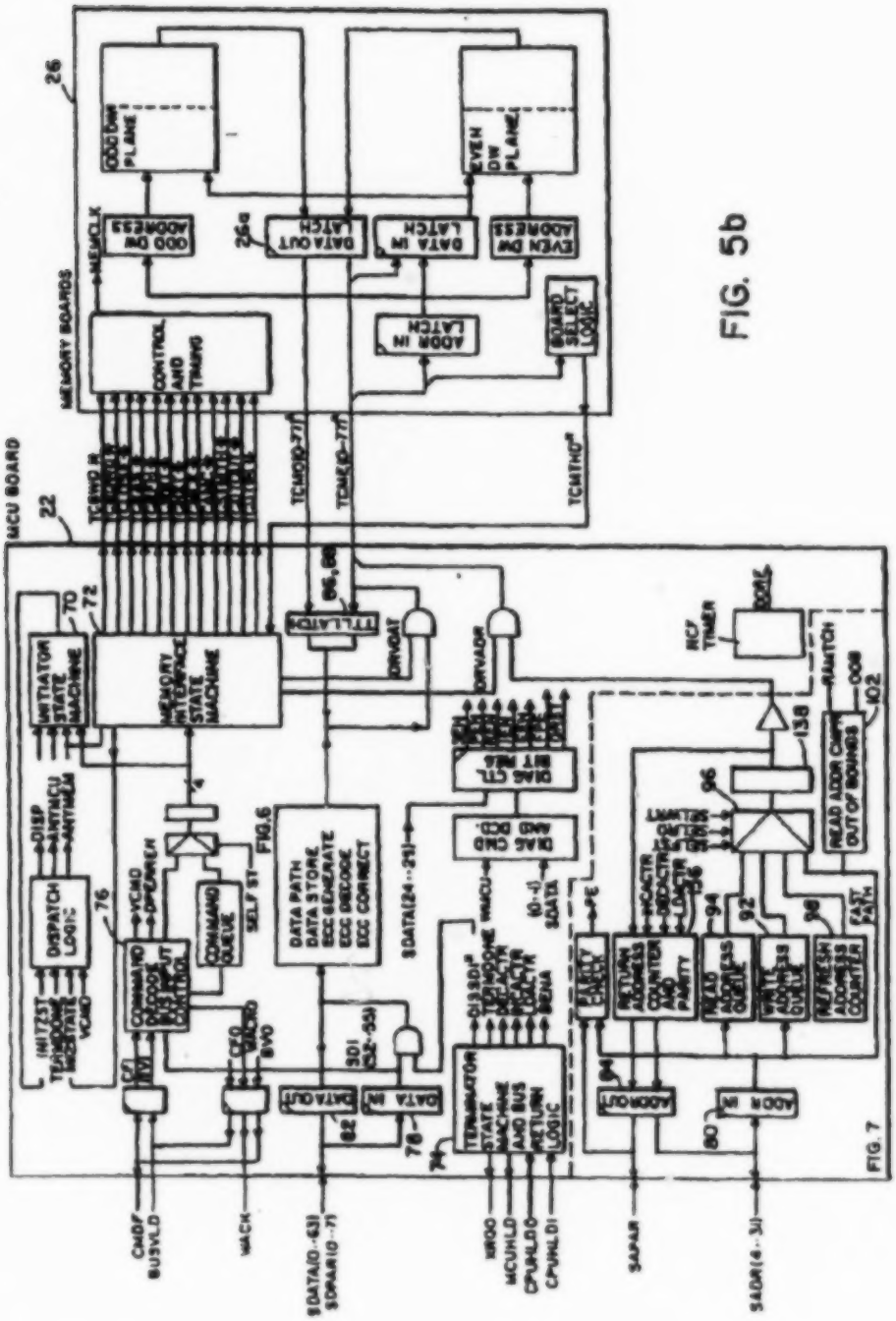

26
MEMORY BOARDS
MCU BOARD
22
CONTROL AND TIMING
DATA OUT LATCH
DATA IN LATCH
ADDR IN LATCH
BOARD SELECT LOGIC
INITIATOR STATE MACHINE
MEMORY INTERFACE STATE MACHINE
DISPATCH LOGIC
COMMAND QUEUE
DATA PATH
DATA STORE
ECC GENERATE
ECC DECODE
ECC CORRECT
TERMINATOR STATE MACHINE
RETURN ADDRESS COUNTER AND PARITY
READ ADDRESS QUEUE
WRITE ADDRESS QUEUE
REFRESH ADDRESS COUNTER
FAST PATH
FIG. 7

FIG. 5b

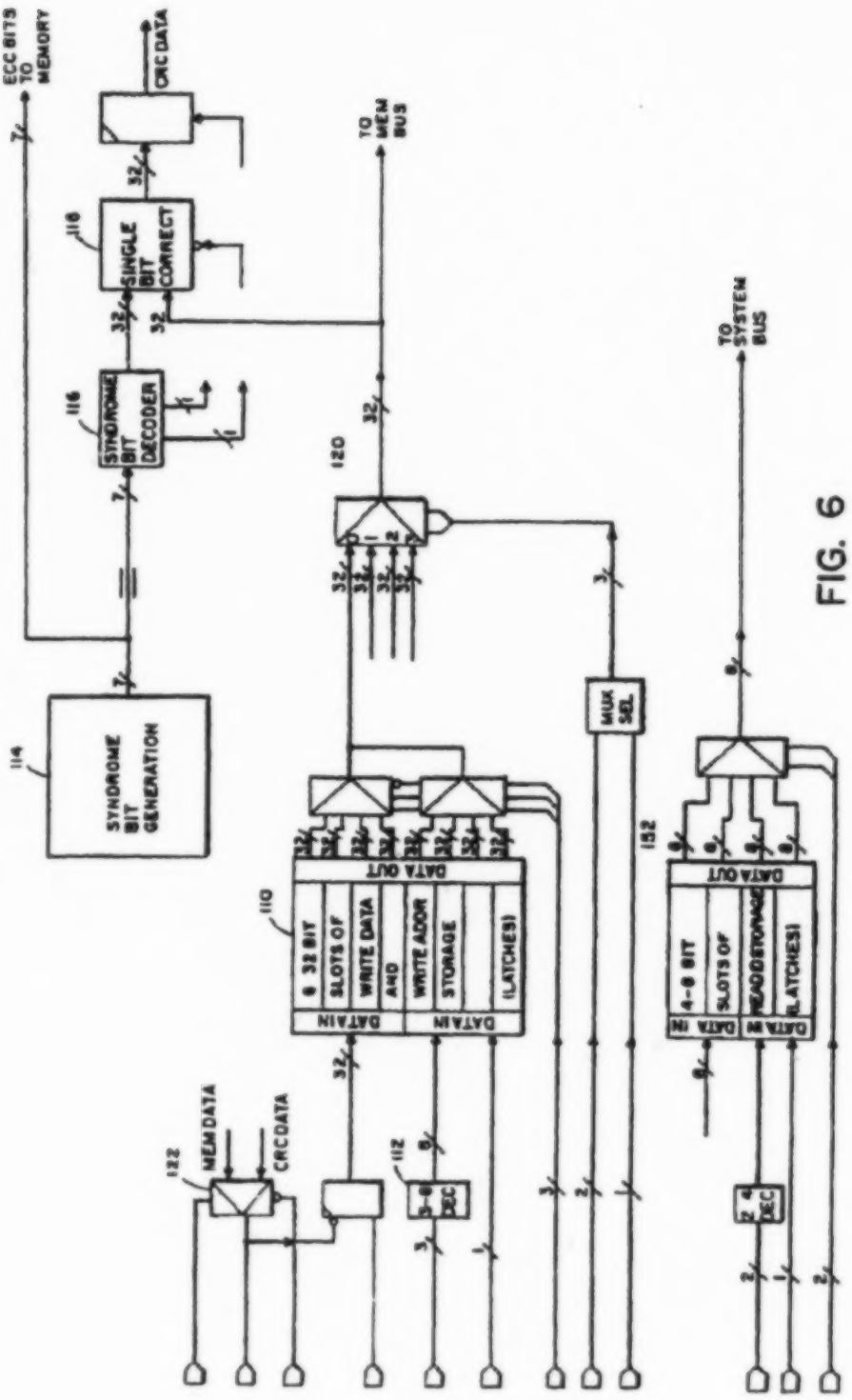

ECC BITS TO MEMORY
CRC DATA
TO MEM BUS
SINGLE BIT CORRECT
SYNDROME BIT DECODER
SYNDROME BIT GENERATION
MUX SEL
TO SYSTEM BUS
DATA OUT
DATA IN
MEM DATA
CRC DATA
DEC

FIG. 6

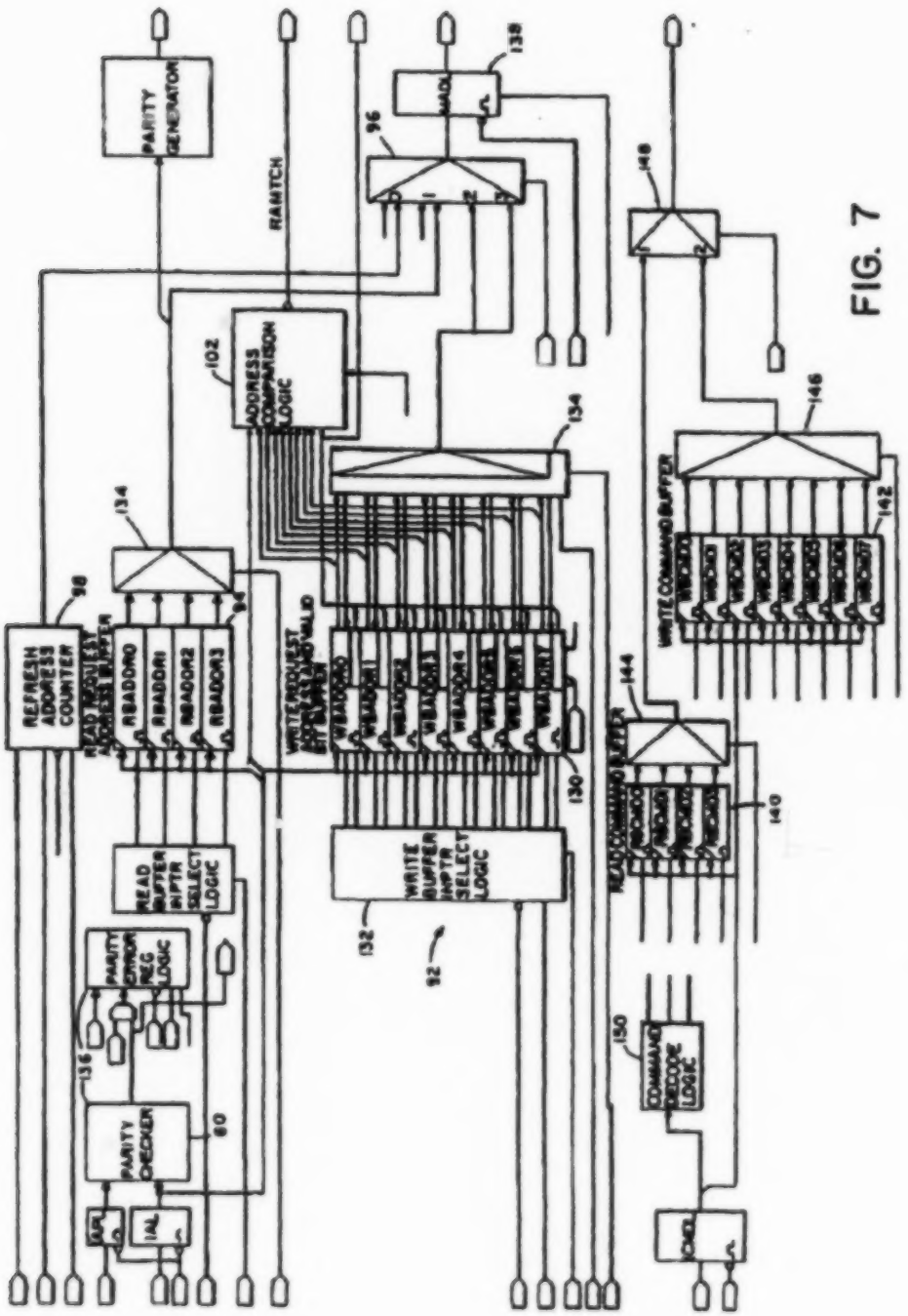
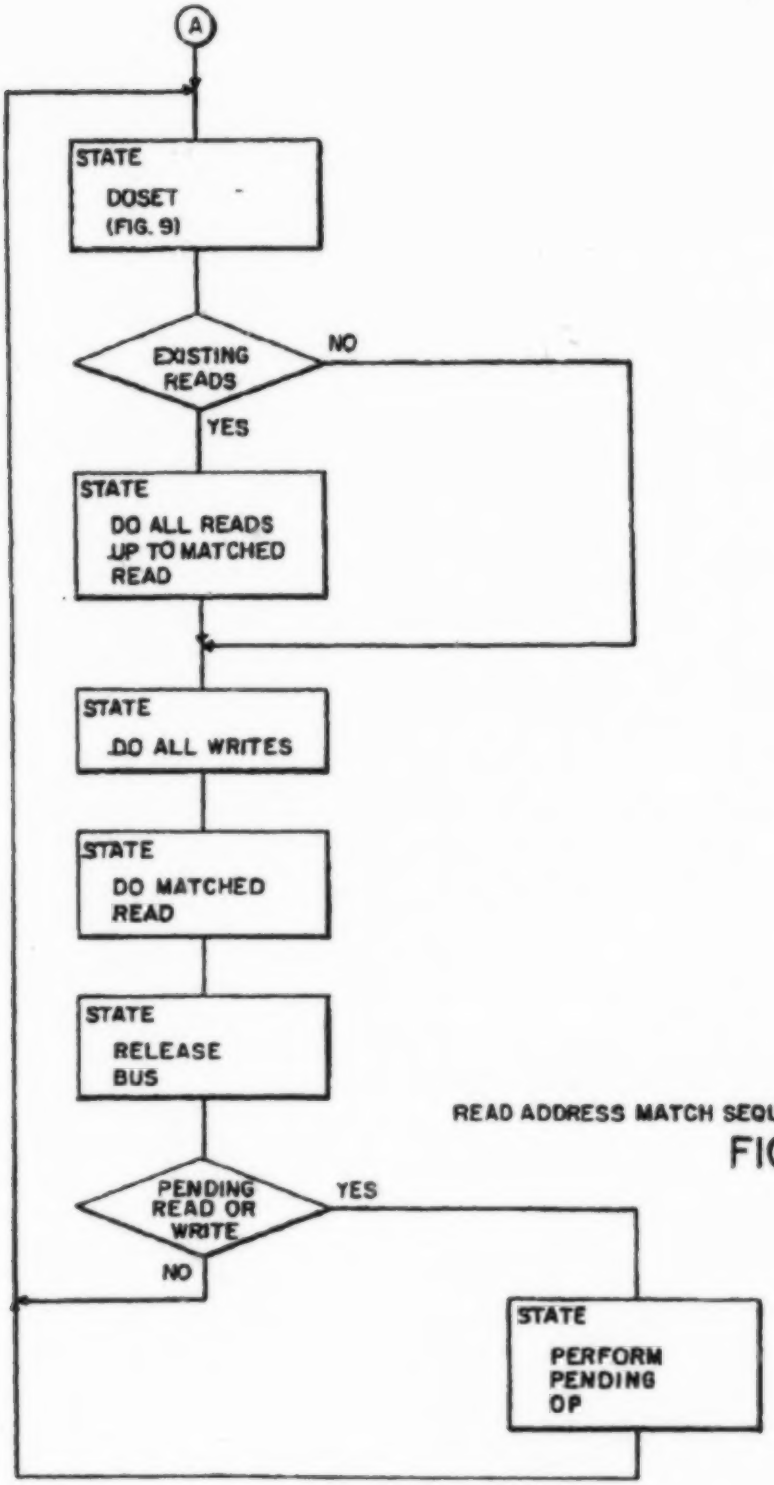
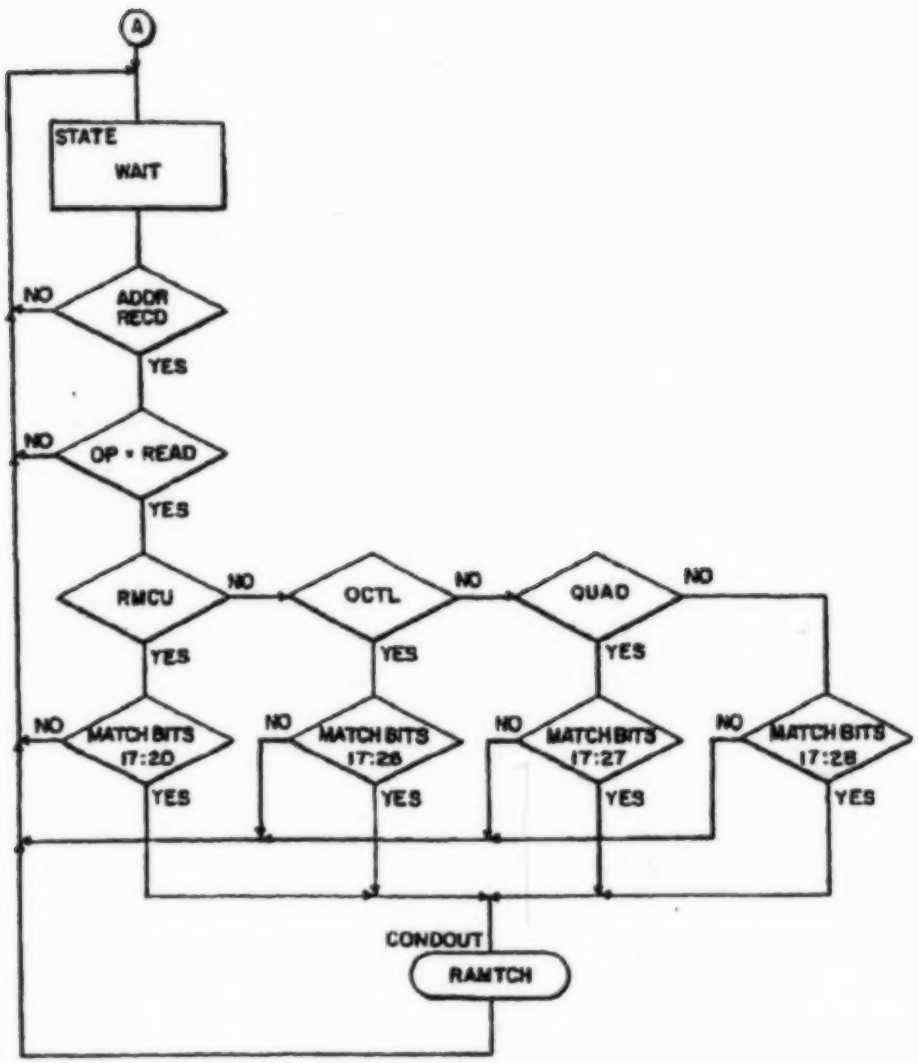

PARITY GENERATION
RAMTCH
ADDRESS COMPARISON LOGIC
REFRESH ADDRESS COUNTER
READ REQUEST ADDRESS BUFFER
RBADDR0
RBADDR1
RBADDR2
RBADDR3
WRITE REQUEST ADDRESS BUFFER
WBADDR0
WBADDR1
WBADDR2
WBADDR3
WBADDR4
WBADDR5
WBADDR6
WBADDR7
WRITE COMMAND BUFFER
WBCMD0
WBCMD1
WBCMD2
WBCMD3
WBCMD4
WBCMD5
WBCMD6
WBCMD7
READ COMMAND BUFFER
RBCMD0
RBCMD1
RBCMD2
RBCMD3
READ BUFFER INPTR SELECT LOGIC
WRITE BUFFER INPTR SELECT LOGIC
PARITY ERROR REG LOGIC
PARITY CHECKER
COMMAND DECODE LOGIC
MUX
98
94
96
102
134
138
148
146
142
144
140
130
132
92
136
80
150

FIG. 7



READ ADDRESS MATCH SEQUENCE

FIG. 8



DOSET SUBROUTINE

FIG. 9

MEMORY CONTROL UNIT WITH SELECTIVE EXECUTION OF QUEUED READ AND WRITE REQUESTS

This is a continuation of copending application(s) Ser. No. 07/213,395 filed on Jun. 30,1988, now abandoned.

FIELD OF THE INVENTION

This invention relates generally to an information processing system and, in particular, to a memory control unit for coupling a system memory to a high speed, non-interlocked system bus.

BACKGROUND OF THE INVENTION:

In an information processing system which employs a central system memory for storing information units, such as operands and instructions, an important consideration is the coupling of the system memory to a central system bus. In order to achieve and maintain a desired system bus bandwidth it is important that data read and write operations which access the system memory occur in a rapid manner. It is also important, for those systems which employ one or more system bus units, or bus connections, that substantially simultaneous multiple read and write accesses which target a given memory location occur such that the read and write operations are correctly sequenced. That is, if one bus connection desires to read a given memory location while another bus connection desires to write the same location the sequencing of these two operations must be such that the data read reflects the current state of the data.

Related to both of the above considerations is an underlying goal of providing the data to a requesting bus connection without the introduction of errors resulting from hard or soft memory storage device errors. In this regard it has been known to provide additional memory storage devices to store error correction and detection syndrome bits which advantageously correct single bit errors and detect multiple bit errors occurring in data words stored within the system memory. It has also been known to provide, for memory systems which employ dynamic random access memories (DRAM), error "sniffing" and "scrubbing" techniques which detect and correct errors during the required memory refresh cycles.

It can be realized however that error detection and correction techniques require some finite amount of time to accomplish. During a read of the system memory this additional time is typically incurred for each memory access, resulting in a longer access cycle and a consequent reduction in the overall bandwidth of the system bus.

SUMMARY OF THE INVENTION

The foregoing problems are overcome and other advantages are realized by an information processing system constructed and operated in accordance with the invention. In accordance therewith a memory control unit comprises first interface means for interfacing the memory control unit to one or more memory units; second interface means for interfacing the memory control unit to a system bus, including a system data bus for expressing information units, including memory read and write requests, and a system address bus; means, responsive to a write request from the system bus, for storing one or more information units within a memory unit at an address specified by the system address bus, the storing means comprising write request receiving and buffer means having a plurality of storage locations for storing received write requests and associated write addresses prior to the execution of the write requests; and means, responsive to a read request from the system bus, for reading one or more information units from a memory unit at a location specified by the system address bus, the reading means comprising read request receiving and buffer means having a plurality of storage locations for storing received read requests and associated read addresses prior to the execution of the read requests. The memory control unit further comprises means, having a first input coupled to the write buffer means and a second input coupled to the read request receiving means, for comparing a received read address to write addresses stored in the write address buffer means, the comparing means having an output for indicating, when asserted, the occurrence of the reception of a read address having a value within a predetermined range of values of one of the stored write addresses.

In accordance with a method of the invention there is disclosed a method of reading and writing the information units which comprises the steps of buffering write requests, including write addresses, as they are received from a system bus; buffering read requests, including read addresses, as they are received from the system bus; and comparing when received each read address against buffered write addresses, if any, to determine if a received read address has a value within a predetermined range of values of a buffered write address. If a received address is not determined to be within the predetermined range of addresses of any buffered write addresses the method further comprises the steps of executing all buffered read requests; and executing all buffered write requests. If a received address is determined to be within the predetermined range of addresses of any buffered write addresses the method further comprises the steps of executing all buffered read requests up to but not including the received read request which was determined to be within the predetermined range of addresses; executing all buffered write requests; and executing the buffered read request which was determined to be within the predetermined range of addresses.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects of the invention will be made more apparent in the following detailed description of a preferred embodiment read in conjunction with the accompanying drawing wherein:

FIG. 1 is a block diagram of an information processing system having an embodiment of the invention;

FIG. 2 is a flow chart representation of a master flow of control of MCU 22;

FIGS. 3a, 3b and 3c are timing diagrams which illustrate the operation of the noninterlocked system bus;

FIGS. 4a–4e illustrates the format of various system IPC commands;

FIG. 5a is a block diagram which conceptually illustrates the MCU 22;

FIG. 5b is a block diagram which shows the implementation of the MCU 22 coupled to a memory array 26;

FIG. 6 is a block diagram which illustrates in more detail a portion of the circuitry of the MCU 22;

FIG. 7 is a block diagram showing in more detail other circuitry of the MCU 22;

FIG. 8 is a flow chart which illustrates the operation of the read address match circuitry; and

FIG. 9 is a flow chart which illustrates the operation of a portion of the associative address comparison as a function of read type.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

Referring to FIG. 1 there is shown an information processing system (SYSTEM) 10 incorporating a preferred embodiment of the invention. As seen, SYSTEM 10 comprises a System Bus (SB) 12 which further comprises a System Address Bus (SA) 14, a System Bus Control Bus (SBCB 15) and a System Data (SD) bus 16. SB 12 functions to convey information units between the various components of the SYSTEM 10. Information units may be addresses, I/O input or output data, operands, instructions or any other type of information which passes between the components of the SYSTEM 10. In the preferred embodiment of the invention SB 12 is a high speed, non-interlocked bus which operates at ECL voltage levels. That is, logic signals on the bus swing between approximately −0.9 volts and −1.8 volts.

Coupled to System Bus 12 are a plurality of system units, or bus connections, which include a first central processing unit (CPU0) 18 and an optional second CPU1 20. Also coupled to SB 12 is a memory control unit (MCU) 22 which has coupled thereto via a memory bus 24 one or more memory boards, such as MEM0 26, MEM1 28 and MEM2 30. In the preferred embodiment of the invention MCU 22 may be coupled to up to eight memory boards, such as the MEM7 32. Each memory board may store from, for example, four million bytes (4 MB) to 128 MB of information depending upon the type and quantity of memory devices installed. SB 12 also has coupled thereto one or more system bus interface (SBI) units, such as the SBI0–SBI3, 34, 36, 38 and 40, respectively. Each of the SBIs is further coupled to an associated I/O data bus (IODB), such as the IODBs 42 and 43 coupled to SBI0 34 and SBI2 38, respectively. Each IODB in turn has coupled thereto up to fifteen intelligent I/O processors (IOPs), such as the IOPs 44–50 and IOPs 45–51. In the preferred embodiment of the invention the IODBs 42 and 43 operate at TTL levels. That is, logic signals on these buses swing between approximately zero volts and +5.0 volts. CPU0 18 and CPU1 20 each comprise an associated high speed cache memory and are each further coupled via a cache data bus (CDB) 52 and 54, respectively, to an arithmetic unit (AU0) 56 and AU1 58, respectively. Also coupled to SB 12 is a support control unit (SCU) 60 having a system console (SC) 62 coupled thereto. A support link bus (SLB) 64 provides access and communications from the SC 62, via SCU 60, to the various units coupled to the SB 12. Diagnostic and other information, such as system initialization data, is generally provided over SLB 64.

In general, the CPUs 18 or 20 generate virtual memory addresses which are translated into physical addresses and issued over SA bus 14. Addresses are received and interpreted by MCU 22 for addressing, via the memory bus 24, instructions and data which are stored in the memory boards 26–32. Data and instructions are read and written over the SD bus 16 in accordance with the information conveyed by SBCB 15. Furthermore, SBIs 34–40 are also operable for transferring memory addresses and data over the system bus 12 for storing and retrieving data from the memory boards 26–32. It should be realized that one or more of the IOPs 44–51 may be coupled to a mass storage device such as a magnetic disk. Also, some of the IOPs may be coupled to data communications means operable for inputting and outputting data from the system 10. The IOPs may also be coupled to operator workstations where an operator enters data into the system.

In order to facilitate the description of the invention the operation of the SB 12 will now be described in further detail.

SB 12 is a synchronous, non-interlocked bus having a 64 bit data path and a 28 bit address path. SB 12 provides a peak interconnect bandwidth of 200 Mb/sec. and is, as previously mentioned, comprised of emitter coupled logic (ECL) drivers and receivers.

The following signals describe the System Bus 12 operation and protocol.

System Data (SDATA(0:63))

System Data Bus 16. All memory data traffic to and from bus connections is transferred via these 64 lines. In accordance with the invention when the CMD Flag, to be described, is asserted then certain of these lines are used to transmit Command-ID information, as described below in relation to CMD and ID.

Data Parity (SDPAR(0:7))

Odd Data Parity. One parity bit for each data byte, eight total, of SDATA 16.

System Address (SA(04:31))

System Address Bus. A bus connection transmits during a memory read or write cycle a memory address to the MCU 22 via these 28 lines. The MCU 22 thereafter drives these lines with the address of data read one bus cycle before the data is driven on the bus. Write-back caches coupled to the SB 12 use the MCU 22 driven address to make directory comparisons to determine if bus intervention is required and also drive the address lines during a cache re-transmission. Also, the system address lines, during an IPC transaction (to be described), convey the IPC message and other IPC related data.

Address Parity (SAPAR)

Odd Address Parity bit.

Command Flag (CMDF)

This line, when asserted by a bus connection, indicates that the SDATA Bus 16 is being used by the bus connection to transmit Command-ID information. When this line is not asserted and the bus is valid as indicated by BUSVLD, described below, CMDF indicates that SDATA 16 is transmitting data.

Command (CMD)

During a bus cycle when CMDF is asserted a bus connection places the type of command on SDATA [48:55]to initiate a memory operation or, in the case of the MCU 22, to return data to a requesting bus connection. The eight bit CMD field encodes the type of bus operation.

In the preferred embodiment of the invention the various types of bus operations encoded by the CMD field are as follows.

OPERATION
No Operation
Read Double Word
Read Quad Word
Read Octal Word

Write Byte
Write Word
Write Double Word
Data Return (transmission from MCU)
Transmit IPC
Read MCU
Write MCU

Other signals which comprise the SB 12 are as follows.

ID Each bus connection has a unique identifier (ID). During a bus cycle when a bus connection asserts CMDF the bus connection drives its unique ID onto SDATA [56:63]along with the bus command (CMD) on SDATA [48:55]. The MCU 22 drives the previously received and buffered ID of a bus connection which made a memory request when the data is returned to the requesting bus connection. During the assertion of a Transmit IPC command on SDATA[48:55]a bus connection drives the ID of the target of the IPC command on SDATA[56:63].

Busy (BSY)

This signal line is asserted by a bus connection during all cycles of a bus operation except the last cycle. BSY is sampled at the end of each bus cycle by all connections wishing to use the bus and indicates, when asserted, that the bus is in use and is unavailable to other bus connections. During a START IO IPC command directed to an IOP through an associated SBI the SBI continues to drive the BUSY line until communication is established with the target IOP and a determination is made as to whether the IOP has accepted the START IO command.

Bus Valid (BUSVLD)

This signal is asserted by a bus connection when valid information is placed on the bus.

Lock (LOCK)

Asserted by a bus connection when it is desired to prevent other bus connections (except the MCU 22) from using the bus. This signal line is utilized to implement semaphore instructions that perform read-modify-write operations.

CPU Hold (CPUHLD)

Two CPUHLD signals are provided, one for each CPU. This signal is generated by a CPU and is sampled at the end of each bus cycle by all other bus connections. This signal indicates that one of the write-back caches may be either retransmitting MCU 22 data or may be updating stored information. CPUHLD has the same effect as BSY; it indicates that the bus is still in use and unavailable to all other bus connections. Like BSY, CPUHLD is deasserted one cycle before the last bus cycle. It is also used by a CPU to interlock fetch/-writeback operations for the cache.

MCU Hold (MCUHLD)

Generated by the MCU and sampled at the end of each bus cycle by all other bus connections. In accordance with one aspect of the invention this signal indicates that the MCU has detected a correctable error and will be retransmitting during the next bus cycle the data in corrected form. The system bus protocol for MCUHLD is similar to that of CPUHLD.

Write Acknowledge (WACK)

Acknowledge generated by the MCU 22 in response to Write operations in the bus cycle following the data cycle and by target devices in response to Interprocessor Communication (IPC) operations.

Target Busy (TB)

This signal is generated by a target device in response to an IPC transmission. TB being asserted indicates that the target is busy and that the transmission was not accepted.

Bus Error (BUSER)

Asserted by any bus connection detecting a bus error.

Mem Exception (MEMX)

Asserted by the MCU 22 during the cycle following address transmission if an Invalid Memory Address is received or during the cycle following data transmission if a double bit, uncorrectable, memory error occurs during a memory read.

Xmit Rq In/Xmit Rq Out (XRQI/XRQO)

This signal is daisy chained between bus connections. A bus connection wishing to use the bus will assert Xmit Rq Out and start transmitting on the next cycle only if the following conditions are met:

Xmit Rq In from its higher neighbor is false;
Busy is False;
Hold is False; and
LOCK is False (Only if not an MCU).

A bus connection passes Xmit Rq In from its higher priority neighbor to Xmit Rq Out which is connected to its lower priority neighbor. In accordance with one aspect of the invention the MCU 22 is assigned to be the highest priority bus connection in the XRQI/XRQO daisy chain. This enables the MCU 22 to gain control of the bus in order to execute queued read and write requests, as will be described in detail below.

The timing diagrams of FIG. 3 illustrate various types of bus transactions including the operation of the multiplexed Command/ID and data path having an associated address which is presented during a bus cycle which precedes the presentation of data on SDATA 16. In the timing diagrams of FIGS. 3a–3c signal timing is referenced to the period of a system clock (CLK), the clock period representing approximately one bus cycle. In a preferred embodiment of the invention the basic timing unit or time interval, that is the period of CLK, is approximately 40 nanoseconds.

FIG. 3a shows a byte/half/word/double write immediately followed by the Command-ID portion of a double word (64 bit) read followed by an MCU data return of the requested double word.

FIG. 3b demonstrates the use of CPUHLD for a cache fetch/writeback. A CPU is shown sending Command-ID information to the MCU 22 for an octal word read and thereafter a double word cache write-back. The MCU 22 responds with the return of four double words. CPUHLD prevents another bus connection from using the bus during this sequence.

FIG. 3c further demonstrates the use of the CPUHLD line. A bus connection is shown requesting a double word read and the MCU 22 returning the requested double word. The cache or caches latch the address of the double word, and do directory look-ups in the following cycle. If a "dirty" match is found by a cache, that cache asserts CPUHLD shortly before the end of the cycle. The CPUHLD line prevents other connections from using the bus until the write-back cache re-transmits the double word along with its address and thereafter releases CPUHLD. BSY is asserted during the first cycle of retransmission and BUSVLD is asserted for two cycles if retransmission is performed.

An Interprocessor Communication (IPC) facility allows bus connections to directly communicate with one other by sending IPC messages. The bus protocol for sending these messages resembles a Write operation except that the Transmit IPC Command is used instead of a Write command. The address that is transmitted

along with the Command ID lines has the format indicated in FIG. 4a. The state of the eight Target field bits specifies a particular target, it being remembered that the ID of the target is conveyed by the ID field of SDATA[56:63].

TARGET

SCU
CPU0
CPU1
SBI0
SBI1
SBI2
SBI3

The 64 bit SDATA bus is used to transmit additional optional message data as required by the various IPC message types. FIGS. 4b, 4c, 4d and 4e illustrate, respectively, an IPC generated by an SBI 34, an IPC generated by a CPU which targets an IOP, an IPC generated by SCU 60 which targets a CPU and an IPC generated by SCU 60 which targets, via an SBI 34, an IOP.

Error Detection

There are four types of error detection mechanisms supported by the SB 12. These four types include:

Data Parity Error Detection: there are eight Data Parity Bits on the 64 bit Data Bus (one parity bit for each byte);

Address Parity Error Detection: there is one Address Parity Bit on the 28 bit Address Bus;

Missing Acknowledge: the Acknowledge control line is used to acknowledge Write and IPC transactions. Read operations are acknowledged by the MCU Data Return Command-ID cycle on the bus; and

Sequence Error: illegal bus control sequences are detected by the bus connections involved in a particular bus transaction.

Bus connections detecting any of the above errors assert the Bus Error line for one system bus clock cycle only. This notifies the SCU 60 of the error. The SCU thereafter retrives the Bus Error signal until the SCU 60 clears the error condition. The bus connection also stores the type of error (errors) in an SCU 60 accessible error register.

As has been previously stated, SB 12 is a noninterlocked bus. A significant reduction in bus cycles made possible by the noninterlocked SB 12 results in greater bus bandwidth made available for peak bus request rates and I/O traffic. Inasmuch as MCU 22 internally queues memory accesses a requesting bus connection is not required to incur the memory access time, thereby freeing the SB 12 for other requesters. If the requested access is a write operation the requester is notified as described above and, if the requested access is a read access, the data is later returned.

Referring now to the flow chart of FIG. 2 in conjunction with FIGS. 5a and 5b there is shown in block diagram form the MCU 22 coupled to a memory array 26 via the memory bus 24. MCU 22 is a buffered memory controller capable of performing read operations of up to an octal word length (32 bytes). The memory array 26 is organized as an odd and an even plane, each plane being of a double word width. As the controller of the system memory, the MCU 22 is operable for accepting requests for service from either the CPU0 or CPU1, the SBI 34 and the SCU 60. As has been previously stated, the MCU 22 may be coupled to up to eight memory arrays.

In general, the MCU 22 logically connects the System Data 16 and Address 14 buses to the memory arrays. In accordance with one aspect of the invention two data paths are provided for memory read operations. A first path is a fast data path which is used when no MCU 22 pending requests are stacked upon an internal queue. The fast path drives uncorrected data directly onto SB 12, the data simultaneously being provided to error detection and correction (ECC) circuitry. If a correctable error is discovered the MCU signals the receiving bus connection and thereafter drives the corrected data onto the bus. If the error is uncorrectable, such as a two bit error, the receiving bus connection is so notified. A second data path is a normal path which is employed when the MCU 22 has pending requests stacked upon the queues or does not otherwise have immediate access to SB 12. Inasmuch as MCU 22 must wait for access to SB 12 to return the requested data, the data is provided to the ECC circuitry before being driven out on to SB 12. It can be appreciated that the probability of an error occurring during any given memory read is relatively small, especially in that MCU 22 normally performs error correction during refresh operations, and that the use of the fast data path provides a significant increase in overall system bus bandwidth.

The MCU 22 comprises a plurality of state machines; including an Initiator State Machine (ISM) 70, a Memory Interface State Machine (MISM) 72 and a Terminator State Machine (TSM) 74. A System Bus Decode (SB DECODE) 76 logic block decodes SB 12 signals and provides outputs which are provided to the ISM 70 and MISM 72. In general, the ISM 70 provides signals which initiate the operation of MCU 22 in response to a read or write request. MISM 72 generates a plurality of output signals, shown generally as the signal MEMORY CONTROL, which is provided to the memory arrays. These signals include memory timing strobes and refresh related signals. The TSM 74 decodes certain of the SB 12 signals and provides other signals to the SB 12 which are utilized when the MCU 22 detects a data error during a data return. The TSM 74 also controls the sequencing of data and address latches used when data and command/ID information units are returned to the SB 12.

In this regard there is provided an input data latch 78, an input address latch 80, and a corresponding pair of output data and address latches 82 and 84, respectively. Between the MCU 22 and the memory arrays there are provided an odd memory plane data latch 86 and an even memory plane data latch 88. Memory addresses are multiplexed onto the even plane signal lines.

Write data from SB 12 is buffered in an octal double word latch, or queue, 90 and associated write addresses are buffered in an octal latch, or queue, 92. Thus, up to eight pending write requests may be buffered by the MCU 26. A quad read address latch, or queue, 94 buffers up four pending read requests. A four input multiplexer 96 selects the source of the address for a particular memory access from either the write address queue 92 or the read address queue 94. Another multiplexer 96 input is provided directly from the input address latch 80 and is employed for a fast memory access path when it is desired to bypass the queues. A refresh address counter 98 provides refresh addresses through multiplexer 96 during memory array refresh cycles. A buffer 100 buffers the selected output of multiplexer 96 onto the even plane signal lines.

In accordance with the invention an address comparison logic block 102 compares an address associated with an incoming read request to write addresses, if any, buffered in queue 92. If a match occurs a signal Read Address Match (RAMTCH) is asserted. RAMTCH being asserted causes normal processing of read and write requests to be suspended and buffered read and write requests to be executed. This aspect of the invention will be described in greater detail below.

MCU 22 also comprises error correction and detection logic 104 which is interposed between the odd and even memory data latches 86 and 88, respectively, and the write queue 90 and the output data latch 82.

In FIG. 2 there is illustrated the flow of control of MCU 22 in response to an MCU operation initiated by a bus connection or by an internally generated refresh request. After exiting an initial reset condition MCU 22 enters an idle state (A) which includes testing for the occurrence of a valid MCU 22 operation (B). If a valid MCU 22 operation is detected a further test is made (C) to determine if the operation is a memory operation, that is an operation which will access one of the memory arrays, or whether the operation is an internal operation. If the latter condition is true a further test (D) is made to determine if the internal operation is a diagnostic operation or an operation that reads or writes data to an internal MCU 22 resource, such as a read or a write to a reference and change table (RCT). If a diagnostic operation is indicated diagnostic control bits are set (E) and the idle state A is entered. If an access to an internal MCU 22 resource is indicated the Initiator State Machine 70 is started (F). A test is performed (G) to determine if the internal operation in progress is an RMCU operation, which requires that a data return be made to SB 12, or is a non-return type of operation, such as a write to the RCT. If the latter is true a test is made (H) to determine if the ISM 70 is idle and a wait state (I) is entered until ISM 70 is finished, at which time the Idle State A is reentered.

If a data return is required the Terminator State Machine 74 is started (J) and a test is made (K), with a wait state (L), until the data return is complete. At the completion of the operation of the TSM 74 the Idle State A is reentered.

At step C, if a memory operation is indicated, the Memory Interface State Machine 72 is started (M) and a test is performed (N) to determine if a read operation, read-modify-write operation, or if a refresh cycle is in progress. If none of these are true it is assumed that a write operation is in progress. Control branches during a write operation to a state (H) where the data is corrected, if required, and the data is written to the memory array. If test N determines that a read, RMW or refresh operation is in progress a test is made (P), with an attendant wait state (Q), for the assertion of a Memory Interface State Machine 72 signal DTOUT. The assertion of DTOUT indicates that a memory array is being requested to provide the data which is currently being indicated by the address being sourced on the even plane data/address signal lines. When the data is returned to MCU 22 a test is made (R) to determine if the data is to be driven onto SB 12 or is to be written back to the memory array (a refresh or RMW access). If the data is to be driven onto SB 12 the flow of control transfers to J where the TSM 74 is started. If the data is to be written to the memory array the flow of control transfers to block 9 where the data is corrected and written. Thereafter a test is made (S) with an attendant wait state (T) to determine if the Memory Interface State Machine 72 is idle, indicating the completion of the write cycle. Flow of control then passes back to the Idle State A.

Referring now to FIG. 6 there is shown in more detail a portion of the MCU 22 circuitry. The circuitry shown in FIG. 6 is, in a preferred embodiment of the invention, embodied in one of a pair of identical gate array devices which are utilized to buffer write data and addresses, buffer read ID data and perform E[illegible] functions. Each write data latch 110 storage location is one word, or 32 bits, in width. A 3-8 decoder 112 provides eight selection outputs for selecting one of the eight latches for storing write data.

BCC circuitry comprises syndrome and write check bit generation logic 114, a syndrome bit decoder 116 and single bit correction circuitry 118. A four input, 32 bit multiplexer 120 selects a source of data for application to the output memory data lines. Input memory data (MEMDAT) is applied to a 2 input multiplexer 122 which permits either the uncorrected memory data or corrected CRCDAT to be output to the system bus latches. During the aforedescribed fast read cycle MEMDAT is output. If an error is detected corrected CRCDAT is subsequently provided.

A four location 8 bit read ID latch 124 is provided for latching a SB 12 bus connection ID during a read access. This latched ID is later driven onto SB 12 during the return of the requested read data in order to identify the target of the data.

Referring now to FIG. 7 there is shown circuitry adapted for buffering read and write request addresses and commands. In a preferred embodiment of the invention the circuitry of FIG. 7 is also embodied in a gate array device.

A Write Request Address and Valid Bit Buffer 130 stores 28 address bits at locations determined by Write Buffer Input Pointer Selection Logic 132. A valid bit associated with each of the write buffer locations is set when an address is stored and is reset when the corresponding write operation is accomplished. One of the write addresses is selected by 8 input multiplexer 134 which supplies the write address to the memory address multiplexer 96. Other inputs to the memory address multiplexer 96 include the refresh address from the refresh address counter 98 and a read address from a four input read address multiplexer 134. Address parity logic 136 detects parity errors which may occur on the input read and write addresses. The output of memory address multiplexer 96 is latched by a memory address output latch 138.

A quad, two bit read command buffer 140 and an octal, two bit write command buffer 142 have outputs which are individually selected by multiplexers 144 and 146, respectively. The outputs of multiplexers 144 and 146 are applied to a read/write command multiplexer 148.

A Command Decode Logic 150 block decodes CMD bits 0, 2 and 3 to determine if an incoming requested read operation is a double word read operation, quad read operation, an octal read operation or a read operation which reads internal data from the MCU 22. These decoded outputs are applied to the Address Comparison Logic 102 block which, as has been described, performs an associative address comparison of all of the buffered write addresses against an incoming read request address. If a match is detected the RAMTCH signal is asserted which initiates special processing of

the read and write buffers. As shown in the flow chart of FIG. 9, depending upon the type of read operation different numbers of bits are compared between the incoming read address and the queued write addresses. For example, during an octal read 10 address bits (17:26) are compared, during a quad read 11 bits (17:27) are compared while a double word read compares 12 address bits (17:28). During an internal read, typically of the RCM, only four bits are compared; these bits corresponding to a memory address page frame associated with the reference and change table.

Due to the noninterlocked operation of the SB 12 the MCU 22 buffers incoming read and write requests, normally allowing incoming reads to proceed ahead of buffered writes. The MCU 22 comprises the address comparison logic 102 to properly sequence the execution of requests in the event that an incoming read address matches, within a predetermined range of memory addresses, a buffered write address. In this case, the queued read and write requests are executed before the read request which caused the match proceeds. The logic employed to perform the match function implements an associative address comparison of all buffered write addresses versus an incoming address associated with a read operation. The least significant twelve bits of the incoming read address are compared down to the double word level, depending on the type of requested operation, that is, on the decoded output of command decode logic 158.

All read requests performed by the MCU 22 are of a double word width or more. The MCU 22 may also perform quad word reads and octal word reads efficiently as the MCU 22 comprises a quad word width incoming data path from the memory boards. During an octal read, the MCU 22 receives two quad words from a memory array, separated in time by three memory clock cycles. This allows the return of data to the requester with a minimum number of wait states once the first double word of data is driven onto the SB 12. Write requests to the MCU 22 may be of byte, word, or double word width. Write requests of less than a word width (byte writes) force a read-modify-write cycle to be executed by the MCU 22.

SB 12 permits a bus connection to send the MCU 22 multiple back-to-back memory write requests. Bus re-arbitration is not necessary during each double word transfer as the SB 12 can be held by the bus connection using the aforedescribed LOCK signal line. This SB 12 LOCK condition may be maintained for up to an octal word write (four time-contiguous write operations). Re-arbitration must be performed, however, after at most every octal word write as the MCU 22 must be capable of requesting and receiving ownership of the system bus after each write transfer.

The MCU 22 has the capability to stop a bus connection after any write request and hold the bus in a quasi-busy state should the Write Buffer fill after a given write request. The same quasi-busy state may be established by the MCU 22 when the Read Buffer fills. This allows the MCU 22 to hold off further bus requests until it can at least partially empty the full buffer. As has been previously stated, the XRQO signal line of SB 12 is used for this purpose. The MCU 22 is defined to be the highest priority device on SB 12 and its XRQO line will force other requesters to hold their bus operations until the MCU de-asserts the XRQO line.

As has been stated, the Write Address is stored within the latch 130 which is used during the associative address compare. This latched storage permits all eight possible queued Write Addresses to be accessible during the associative address compare. The address compare is performed when a Read operation is decoded from the CMD (0:3) lines which are been latched from the System Bus 12. As has also been stated, the associative compare generates a bit, called the RAMTCH (Read Address Match) bit. This bit being asserted serves as a flag which indicates that special sequencing is required by the MCU 22. The RAMTCH bit being asserted causes the bus interface controller to assert XRQO to lock SB 12 to prevent the entry of a further Write request into the Write Buffer. Then, no further Write requests are accepted from the System Bus 12 until the MCU 22 has cleared the RAMTCH flag by performing a RAMTCH operation sequence. The RAMTCH operation sequence is illustrated in the flow chart of FIGS. 8 and 9.

The assertion of the RAMTCH flag indicates that a requested read operation will read a location which has a write request pending as well. The write request should preferably proceed ahead of this read request otherwise old data may be returned to the requester and data integrity may not be preserved. In accordance with one aspect of the invention the requests queued in Read Buffers 94 and 140 are executed to the point where the Read request which caused the assertion of the RAMTCH bit is encountered. Before this Read operation can proceed, the requests queued in Write Buffers 90 and 92 are executed, including the queued Write request which triggered the assertion of RAMTCH. The Read operation is then accomplished and the data returned to the requester reflects the result of the just executed Write operation. SB 12 is then released so further system bus activity may occur.

It should be noted that the assertion of the RAMTCH signal indicates that the Write Buffer contains at least one, and possibly more, write operations that must complete before the read request which caused the assertion of RAMTCH can start. As has been stated, the state of the Write buffer contents must be frozen at this point. This is done to the fact that all pending Write requests are executed before the execution of the Read request which caused the match. If another write operation were allowed to enter the Write Buffer, this operation may have as a target the same location that the matched Read operation desires. If the data from this later Write operation is allowed to enter the system memory during the flush of the Write Buffer, the Read operation may read data which is too "new" for it. Thus, further writes must be halted until the read which set the RAMTCH completes. The following illustrate memory request sequence illustrates the foregoing description of the operation of MCU 22.

SEQUENCE OF REQUESTS TO MCU

Read A
Read B
Write X
Read X
Write Y
Write X

If it is assumed at the start of the foregoing sequence that no requests are pending in the Write Buffer and the Read Buffer then Read A and Read B will proceed normally, that is, the RAMTCH bit will not be asserted. Write X is placed into the Write Buffer to be written to the system memory when the Read Buffer is empty.

Read X then enters the MCU 22 and forces the assertion of the RAMTCH bit due to the associative address compare of the incoming Read X address with the address of Write X. The assertion of RAMTCH also causes the MCU 22 to lock the System Bus 12 to prevent the entry of further Write requests. Meanwhile, Write Y has entered the System Bus Latch on the MCU 22. Write Y is the last operation that can enter the MCU 22 until the MCU 22 releases the bus. However, the Write Y request is not processed until the read requests within the Read Buffer are executed. The Write Y request is placed into the Write Buffers and is specially marked, as will be described, so that it will not commence until the previous Read X operation has completed.

Inasmuch as no further requests can now be received from SB 12 the MCU 22 completes the Read B operation and then executes the pending requests in the Write Buffer, due to the RAMTCH bit being asserted for the Read X operation. The MCU 22 thereafter completes the Read X operation. At the completion of this operation, the system Bus 12 is released by the MCU 22 so that the Write Y operation can commence and the Write X operation can enter the MCU 22 and execute in its turn.

Were the SB 12 not halted at the Write Y operation, the second Write X operation could enter the Write Buffer before the execution of the Read X. After Write Buffer request execution, the data from the second Write X operation would be read by the Read X operation and erroneous data may be returned to the requester.

A Read request entering the MCU 22 which finds no queued Read requests is directed immediately to the memory arrays for execution. This allows a minimum latency for a read request from the requester's viewpoint. Write requests entering the MCU 22 are immediately buffered unless the MCU 22 is in the Idle State (FIG. 2), whereupon the execution of the Write request is immediately begun. The buffering of Write requests allows Write requests to enter the MCU 22 while a Read operation is in progress. Read and Write requests may thus enter the MCU 22 while other operations are in execution so long as the RAMTCH flag is not asserted. The operation of MCU 22 then proceeds, allowing reads to execute and writes to be buffered while the reads are executing. At a point in time where all Read requests have been executed and no further Read requests are incoming from the System Bus 12, the execution of queued Write requests, if any, begins. If there are no queued Read or Write requests and no incoming operation from SB 12, the MCU 22 is considered to be in the Idle State.

During read operations, if a predetermined number of Write requests become queued ownership of SB 12 is requested by the MCU 22. All buffered Read requests are then executed and at least one Write request is executed before the MCU 22 again releases SB 12. The same sequence is followed if a predetermined number of Read requests accumulate. This prevents loss of memory operations due to buffer overflow. As the MCU 22 is the highest priority device on the System Bus 12 during normal operation (it is first on the XRQ0/XRQ1 daisy chain), it can effectively become the SB 12 master at any time. However, inasmuch as the SB 12 is essentially pipelined, one further memory operation may enter the MCU 22 System Bus Latch before the MCU 22 gains bus control. This further memory operation, as was previously mentioned, is marked until such time as the MCU 22 can again allow a normal flow of requests to proceed.

The predetermined number of pending Write requests is preferably set at four double words due to the operation of the write-back cache memory associated with CPU0 18 and CPU1 20. That is, the MCU 22 must be ready to accept at anytime a cache write-back of four double words. Thus, by maintaining a maximum number of pending Write requests of four a cache write-back may be accommodated at anytime.

During memory refresh, all memory array boards are refreshed simultaneously. In addition, data "sniffing" and "scrub" cycles are performed at this time, with one double word being read back from one bank of the dynamic RAM memory. Thus, the refresh cycle is similar to an extended double word read. If an error is detected, the error is corrected and the corrected data is returned to the memory arrays. During a normal double word read, only one division of a memory array (upper or lower) receives the RAS and CAS strobes while the other division remains in a standby mode (no RAS, no CAS). During refresh, both divisions receive RAS simultaneously. During the aforementioned storage of a corrected double word the memory array division receiving the corrected data will further have its associated CAS strobe asserted.

A refresh cycle normally requires six memory clock cycles to complete, assuming that no error is detected. If an error is detected and is correctable, the refresh cycle requires an additional write double word access time of five cycles.

The sniff logic corrects and writes back any single bit errors. This has the effect of periodically scrubbing the memory clean of single bit errors. Each time a refresh operation is performed, a double word of data is read and, if necessary, corrected and written back to the location from which it came. Refresh addressing logic contains up to 10 bits for the RAS portion of the address and 15 address bits to be divided between the CAS portion of the address, decoding logic to determine a memory board to be selected for a read, and bank decode logic to select the bank and division (upper/lower) to be read.

The MCU 22 has a number of paths through which address information can flow. The normal path is for the address to enter through the Input Address Latch 80 and then into the Write/Read Queues 92 and 94. Once placed into the Write/Read Address Queues 92, 94, the address remains there until that particular write operation is called up for execution. In the case of an idle MCU, the address effectively bypasses the queues and is immediately applied, or set up, to the memory, therefore expediting read and write operations.

Data that is to be written to the main memory is passed through the 4 to 1 multiplexer 120. The output of multiplexer 120 is provided to several destinations. One of these destinations is directly to the memory arrays. Another destination is the ECC encode/decode logic where the check bits are generated and sent directly to memory with the data on a memory write operation. The data and check bit signal lines are bi-directional. This allows data and check bit output during a write operation. During a memory read, the direction is reversed.

A more complex data path exists for read-modify-write operations. Such operations require that a double word of data be read from a memory unit and that the

data be checked for errors. If an error occurs the error is corrected and the corrected data is merged with a byte of data which has been stored in the Write Buffer 118. This is accomplished using the multiplexer 120 and selecting, on a byte basis, the source for the multiplexer to be either the latched and corrected read data or the Write Buffer output. The merged data is thereafter re-encoded and sent to the memory by the same path as a normal write operation.

The ID of a bus connection to which data is to be returned is buffered in Read ID storage latches 152 and is multiplexed onto the SB 12 data path during the MCU 22 data return cycle. During a read operation, while the MCU 22 awaits the return of data from a memory array, MCU 22 sets up the return ID and address for the SB 12 Data Return cycle. One cycle before the data is sent over SB 12, the requester's ID, a Data Return Command, and the address originally provided by the requester are returned to the requester. If any delay is incurred in obtaining ownership of the system bus, the Data Return cycle is delayed. Resource conflicts are avoided by use of the latches 82 and 84, where the return Command, ID and Address are stored until driven onto the bus.

If a correctable error occurs, the data path multiplexer 122 (a 2 to 1 multiplexer) is flipped. One side of this multiplexer 122 receives inputs directly from the memory. The other side receives latched, corrected data from the output of the ECC circuit. Thus, if an error is detected, the corrected data can be sent rapidly by flipping the data output multiplexer 122. At the same time the corrected data is being returned to the requester, the next double word of data (if a multiple double word transaction) is being latched for transmission during the next cycle. If it too is in error it will be sent and then corrected during the following cycle and re-transmitted as was the first double word.

The MCU 22 of the invention also provides correct data when more than one double word of a multi-double word read request is in error. In this case latches back to those on the output of the memory arrays can be used to hold the data until the MCU 22 can process it. A four-stage pipeline is provided to accommodate all possible cases of erroneous data appearing in double words. The four stages comprise latches 24a on the output of the memory arrays (FIG. 5b); latches 82 and 84 at the input of the MCU; and latches 104a and 104b (FIG. 5a) at the input and output, respectively, of the ECC logic 104.

The MCU 22 has both a fast and an internal address path. Whenever the MCU 22 is in the Idle State (not processing any requests with none stored in either the read or write buffers), an incoming operation (either read or write) is begun immediately. The address path is steered toward the fast path. This allows a direct path from the system bus input latches to the drivers on the edge of the MCU 22. If the operation is an MCU internal function (Read or Write MCU), the MCU 22 will begin the operation immediately and will respond (if data is to be sent to the requester) within two cycles after the request enters the MCU. The address will be sent to the memory arrays but the row address strobe (RAS) signal, which initiates all memory operations, is not generated.

The internal path is used if an incoming request must await completion of a previously buffered operation. This path takes the address from the system bus holding latch directly into either the Read Buffer Address latches or the Write Buffer Address latches, 94 and 130, respectively.

An address parity check is performed when addresses enter the MCU 22 as well as an Out of Bounds check. If an error is detected, the BUSERR line is asserted to so indicate. If an address error is detected when a request enters the MCU 22, the MCU will not buffer the operation and will not execute it. It is as if the operation were never sent except that the MCU must notify the SCU 60 of the parity error on the address. In the case of the Out of Bounds check being true, the requested operation has a target location outside the region of available memory, the MCU 22 will assert the MEMX signal line two cycles after the address cycle and will ignore the operation. This condition will be reported to the SCU 60.

When a read operation is performed, the return address is inputted into the latch 84 for return to the requester. The Physical Frame Bits (address bits 04:20) are held in a secondary latch. The page offset bits down through bit 28 (address bits 21:28) are passed through an up/down counter 186 so that the base address may be incremented by the MCU 22 for multi-double word returns. The counter requires up/down capability due to the need to send corrected data after the incorrect data is initially sent (this is during a correctable error sequence). LSB address signals 29:31 are zeroed before being returned from the MCU 22. Parity is generated once on address bits 04:20 and is re-generated over bits 21:28 and is then combined with the upper bit address parity every cycle during a multi-double word data return.

Due to the high speed of SB 12 the MCU 22 decodes the ECC error bits on the fly, as it is sending the data back to the requester. The fast data path actually bypasses the error correction logic and data is returned to the requester, even though it may contain an error. If it does contain an error, the data must be corrected and re-transmitted to the requester during the following cycle. In order to accomplish this function the SB 12 signal line MCUHOLD is provided. MCUHOLD is driven by the MCU to all other bus connections in order to notify the other bus connections of the MCU's intention to retain ownership of SB 12 even though it has released BUSY. MCUHOLD, in turn, is driven, at least in part, by the Single Bit Error Detect logic.

When the check bits are encoded and are sent to memory, the properties of the code which produce them are that exactly three copies of each of 32 data bits go into the formation of seven check bits per each word in a double word. When the check bits return from memory, each one is exclusive ORed with the particular data bits that corresponded to each check bit when generated. Thus, the information contained within these seven bits determines whether there are no bits in error, whether there is a single bit in error, or whether there are multiple bits in error. A single bit error is correctable while multiple bits in error are not. If the seven check bits are all logical zeroes, then there are no bits in error. If there is a single bit in error, then three of the seven check bits will be a logical one. An even number of ones set within the seven check bits indicates that there are at least two bits in error. In the case of a single-bit error, the parity of the check bits alone will be odd. Thus, a simple way to determine quickly that a single-bit error exists by determining the parity of the seven check bits.

The occurrence of a double bit error need not be indicated until one cycle after the data has been re-

turned. To indicate the occurrence of a double bit error the MEMX signal line is pulsed high. The address having an uncorrectable error and the associated syndrome pattern is reported to the SCU 60 via the SLB 64.

The invention described above may be embodied in yet other specific forms without departing from the spirit or essential characteristics thereof. Thus, the present embodiments are to be considered in all respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than by the foregoing descriptions, and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced therein.

What is claimed is:

1. A memory control unit for controlling a main system memory of a data processing system, the main system memory being comprised of at least one memory unit, comprising:

first interface means for coupling said memory control unit to the at least one memory unit of the main system memory;

second interface means for coupling said memory control unit to a system bus having signal lines for expressing information units, including memory read and write requests, the system bus including a system address bus;

means, coupled to said first and to said second interface means and responsive to a write request from said system bus, for executing the write request by storing one or more information units within a memory unit at an address specified by the system address bus, said write request executing means comprising write request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received write requests and associated write addresses prior to execution of the write requests;

means, coupled to said first and to said second interface means and responsive to a read request from said system bus, for executing the read request by reading one or more information units from a memory unit at a location specified by the system address bus, said read request executing means comprising read request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received read requests and associated read addresses prior to execution of the read requests;

said memory control unit further comprising:

means, having a first input coupled to said write buffer means and a second input coupled to said read request receiving means, for comparing a received read address to write addresses stored in said write address buffer means, said comparing means having an output signal for indicating, when asserted, an occurrence of the reception of a read address within a predetermined range of addresses of one of said stored write addresses; and

means for controlling the execution of read and write requests, said controlling means being coupled to said comparing means output signal and being responsive to said comparing means output signal not being asserted for causing an execution of all buffered read requests before any buffered write requests, said controlling means further being responsive to said comparing means output signal being asserted for first causing an execution of only those buffered read requests which precede a buffered read request which caused the assertion of said comparing means output signal and then causing an execution of buffered write requests.

2. A memory control unit as defined in claim 1 wherein said controlling means further comprises means coupled to said second interface means and responsive to the assertion of said comparing means output signal for preventing a reception of any further read or write requests from said system bus until the execution of all buffered write requests.

3. A memory control unit as defined in claim 1 and further comprising:

means, coupled to said second interface means and responsive to an execution of a read request, for returning one or more information units to said system bus; and

error detection and correction means for detecting and correcting a single bit error in an information unit returned to said system bus, thereby generating a corrected information unit, and including means for subsequently providing the corrected information unit to said returning means.

4. A memory control unit as defined in claim 3 wherein:

said second interface means comprises means for requesting and receiving access to said system bus; and

said returning means is responsive to the operation of said second interface means for returning an information unit to said system bus prior to a completion of the operation of said error detection and correction means, said returning means being coupled to said error detection and correction means for receiving a corrected information unit therefrom for returning the corrected information unit to said system bus at a time subsequent to a time when an information unit having a single bit error is returned to said system bus.

5. A memory control unit as defined in claim 4 wherein said returning means comprises means for indicating on said system bus that an information unit being returned has a bit in error.

6. A memory control unit as defined in claim 3 wherein said error detection and correction means further comprises means for detecting multiple bit errors in an information unit returned to said system bus, and wherein said returning means further comprises means for indicating on said system bus that an information unit being returned has more than one bit in error.

7. In an information processing system having a system bus for coupling together a plurality of bus connections, one of the bus connections being a memory control unit coupled to one or more memory units, the memory control unit being responsive to address and data signal lines of the system bus for writing information units to and for reading information units from the memory units, a method of reading and writing the information units comprising the steps of:

buffering write requests, including write addresses, as they are received from the system bus;

buffering read requests, including read addresses, as they are received from the system bus; comparing when received each read address against buffered write addresses, if any, to determine if a received read address has an address value within a prede-

termined range of address values of a buffered write address;

if a received address is determined not to be within the predetermined range of addresses of any buffered write addresses then:

first executing in sequence all buffered read requests; and

then executing in sequence all buffered write requests;

else if a received address is determined to have an address value within the predetermined range of address values of any buffered write address:

first executing in sequence all buffered read requests up to but not including the received read request which was determined to be within the predetermined range;

then executing all buffered write requests; and

then executing the buffered read request which was determined to be within the predetermined range.

8. The method of claim 7 wherein the first step of executing when a received read address is determined to be within the predetermined range of addresses includes an initial step of asserting a system bus signal which prevents other bus connections from issuing any further read requests or write requests.

9. The method of claim 7 wherein the steps of executing in sequence all buffered read requests are accomplished by:

applying a read address to the memory units;

retrieving an information unit from a memory unit storage location corresponding to the applied read address;

applying, during a first interval of time, the retrieved information unit to an error detection and correction unit to determine if the retrieved information unit has at least one bit in error, the error detection and correction unit further including means for correcting a single bit error in the retrieved information unit; and

driving, during the first interval of time, the retrieved information unit onto the system bus data lines.

10. The method of claim 9 wherein if the error detection and correction unit indicates that the retrieved information unit has a bit in error further comprises the steps of:

asserting a system bus signal for indicating that the retrieved information unit has a bit in error; and

driving during a second, later interval of time the corrected information unit onto the system bus data lines.

11. The method of claim 9 wherein if the error detection and correction unit indicates that the retrieved information unit has more than one bit in error further comprises a step of asserting a system bus signal for indicating an error condition.

12. The method of claim 11 wherein if a received read or write address has a value which exceeds a predetermined maximum value further includes a step of asserting the system bus signal that indicates the error condition.

13. A memory control unit for controlling a main system memory of a data processing system, the main system memory being comprised of at least one memory unit, comprising:

first interface means for coupling said memory control unit to the at least one memory unit of the main system memory;

second interface means for coupling said memory control unit to a system bus having signal lines for expressing information units, including memory read and write requests, the system bus including a system address bus;

means, coupled to said first and to said second interface means and responsive to a write request from said system bus, for executing the write request by storing one or more information units within a memory unit at an address specified by the system address bus, said write request executing means comprising write request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received write requests and associated write addresses prior to execution of the write requests;

means, coupled to said first and to said second interface means and responsive to a read request from said system bus, for executing the read request by reading one or more information units from a memory unit at a location specified by the system address bus, said read request executing means comprising read request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received read requests and associated read addresses prior to execution of the read requests;

said memory control unit further comprising:

means, having a first input coupled to said write buffer means and a second input coupled to said read request receiving means, for comparing a received read address to write addresses stored in said write address buffer means, said comparing means having an output signal for indicating, when asserted, an occurrence of the reception of a read address within a predetermined range of addresses of one of said stored write addresses, the comparing means output being coupled to means for controlling an order of execution of stored read requests and stored write requests as a function of whether the comparing means output signal is asserted or is not asserted; and

means for decoding said read requests to determine a type of read request, said decoding means having an out coupled to said comparing means for determining a number of bits of said read address which are compared to the buffered write addresses, the number of bits being determined as a function of the type of read request.

14. A memory control unit as defined in claim 13 wherein said decoding means is responsive to read requests for determining if one of said read requests is associated with a first type of request which reads data from a location within said memory control unit, a second type of read request which reads an octal word of data from a memory unit, a third type of read request which reads a quad word from one of a memory unit, or a fourth type of read request which reads at least a double word from a memory unit.

15. A memory control unit as defined in claim 14 wherein said comparing means compares four bits when said decoding means determines that a read request is associated with said first type of read request.

16. A memory control unit as defined in claim 14 wherein said comparing means compares ten bits when said decoding means determines that a read request is associated with said second type of read request.

17. A memory control unit as defined in claim 14 wherein said comparing means compares eleven bits when said decoding means determines that a read request is associated with said third type of read request.

18. A memory control unit as define in claim 14 wherein said comparing means compares twelve bits when said decoding means determines that a read request is associated with said fourth type of read request.

19. A memory control unit for controlling a main system memory of a data processing system, the main system memory being comprised of at least one memory unit, comprising:

first interface means for coupling said memory control unit to the at least one memory unit of the main system memory;

second interface means for coupling said memory control unit to a system bus having signal lines for expressing information units, including memory read and write requests, the system bus including a system address bus;

means, coupled to said first and to said second interface means and responsive to a write request from said system bus, for executing the write request by storing one or more information units within a memory unit at an address specified by the system address bus, said write request executing means comprising write request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received write requests and associated write addresses prior to execution of the write requests;

means, coupled to said first and to said second interface means and responsive to a read request from said system bus, for executing the read request by reading one or more information units from a memory unit at a location specified by the system address bus, said read request executing means comprising read request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received read requests and associated read addresses prior to execution of the read requests;

said memory control unit further comprising:

means, having a first input coupled to said write buffer means and a second input coupled to said read request receiving means, for comparing a received read address to write addresses stored in said write address buffer means, said comparing means having an output signal for indicating, when asserted, an occurrence of the reception of a read address within a predetermined range of addresses of one of said stored write addresses, the comparing means output being coupled to means for controlling an order of execution of stored read requests and stored write requests as a function of whether the comparing means output signal is asserted or is not asserted; and

means, having an input coupled to said comparing means output signal and an output coupled to said system bus for asserting a signal on said system bus when said comparing means output signal is asserted, said signal preventing a reception of further read or write requests from said system bus.

20. A memory control unit for controlling a main system memory of a data processing system, the main system memory being comprised of at least one memory unit, comprising:

first interface means for coupling said memory control unit to the at least one memory unit of the main system memory;

second interface means for coupling said memory control unit to a system bus having signal lines for expressing information units, including memory read and write requests, the system bus including a system address bus;

means, coupled to said first and to said second interface means and responsive to a write request from said system bus, for executing the write request by storing one or more information units within a memory unit at an address specified by the system address bus, said write request executing means comprising write request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received write requests and associated write addresses prior to execution of the write requests;

means, coupled to said first and to said second interface means and responsive to a read request from said system bus, for executing the read request by reading one or more information units from a memory unit at a location specified by the system address bus, said read request executing means comprising read request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received read requests and associated read addresses prior to execution of the read requests;

means, having a first input coupled to said write buffer means and a second input coupled to said read request receiving means, for comparing a received read address to write addresses stored in said write address buffer means, said comparing means having an output signal for indicating, when asserted, an occurrence of the reception of a read address within a predetermined range of addresses of one of said stored write addresses, the comparing means output being coupled to means for controlling an order of execution of stored read requests and stored write requests as a function of whether the comparing means output signal is asserted or is not asserted; and

said memory control unit further comprising:

means for returning, in response to an execution of a read request, at least one information unit to said system bus;

means, having an input coupled to said read request executing means, for determining during a time that an information unit is returned to said system bus if one or more bits of the information unit read from a memory unit is in error, the determining

means further including means for correcting a single bit error in an information unit for providing to said returning means a corrected information unit to be subsequently returned to said system bus; and

means, having an input coupled to said determining means and an output coupled to said system bus, for asserting a first signal on said system bus when said determining means determines that an information unit has one bit in error.

21. A memory control unit as defined in claim 20 wherein said asserting means is operable for asserting a second signal on said system bus, said second signal

being asserted when said determining means determines that an information unit has more than one bit in error.

22. A memory control unit for controlling a main system memory of a data processing system, the main system memory being comprised of at least one memory unit, comprising:

first interface means for coupling said memory control unit to the at least one memory unit of the main system memory;

second interface means for coupling said memory control unit to a system bus having signal lines for expressing information units, including memory read and write requests, the system bus including a system address bus;

means, coupled to said first and to said second interface means and responsive to a write request from said system bus, for executing the write request by storing one or more information units within a memory unit at an address specified by the system address bus, said write request executing means comprising write request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received write requests and associated write addresses prior to execution of the write requests;

means, coupled to said first and to said second interface means and responsive to a read request from said system bus, for executing the read request by reading one or more information units from a memory unit at a location specified by the system address bus, said read request executing means comprising read request receiving and buffer means having a plurality of storage locations capable of storing a plurality of received read requests and associated read addresses prior to execution of the read requests;

said memory control unit further comprising:

means responsive to the operation of said write request receiving and buffer means and to said read request executing means for determining, at least during a time that said read request executing means is executing a buffered read request, when a predetermined number of write requests are buffered within said write request buffer means, the predetermined number being less than a maximum possible number of buffered write requests;

means responsive to the operation of said determining means for causing said second interface means to assert a signal on said system bus when said determining means determines that the predetermined number of write requests are buffered within said write request buffer means, an assertion of said signal preventing a reception of further read or write requests from said system bus; and

means responsive to the operation of said determining means for executing at least one buffered write request such that the remaining number of buffered write requests is less than the predetermined number.

23. A memory control unit as set forth in claim 22 and further comprising means, having a first input coupled to said write buffer means and a second input coupled to said read request receiving means, for comparing a received read address to write addresses stored in said write address buffer means, said comparing means having an output signal for indicating, when asserted, an occurrence of the reception of a read address within a range of addresses of one of said stored write addresses, said memory control unit further comprising means for controlling the execution of read and write requests, said controlling means being coupled to said comparing means output signal and being responsive to said comparing means output signal not being asserted for causing an execution of all buffered read requests before any buffered write requests, said controlling means further being responsive to said comparing means output signal being asserted for causing an execution of only those buffered read requests which precede a buffered read request which caused the assertion of said comparing means output signal and then causing an execution of buffered write requests.

24. A memory control unit coupled between a system bus and a main system memory for controlling the operation of the main system memory, the system bus being comprised of address signal lines for specifying addresses within the main system memory, data signal lines for conveying data units between bus connections coupled to the system bus and the main system memory, and at least one signal line for specifying whether a particular system bus transaction initiated by one of the bus connections is a main system memory read request or a main system memory write request, the memory control unit comprising:

means coupled to the system bus for receiving main system memory read requests and main system memory write requests therefrom;

means coupled to the main system memory for executing main system memory read requests and main system memory write requests;

means coupled to the receiving means for buffering a plurality of main system memory write requests, including a main system memory write address associated with each main system memory write request, prior to execution of main system memory write requests;

means coupled to the receiving means for buffering a plurality of main system memory read requests, including a main system memory read address associated with each main system memory read request, prior to execution of main system memory read requests;

means having a first input coupled to the receiving means and a second input coupled to the means for buffering main system memory write requests for comparing a received main system memory read address against all buffered main system memory write addresses, if any, to determine if a received main system memory read address has an address value within a predetermined range of address values of one of the buffered main system memory write addresses, the comparing means including means for asserting an output signal for indicating that a received main system memory read address has an address value within the predetermined range of address values of one of the buffered main system memory write addresses; and wherein

the executing means has an input coupled to the comparing means output signal and is responsive to an assertion thereof for first executing in order as received all buffered main system memory read requests up to but not including the main system memory read request which was determined to be within the predetermined range of address values, then executing in order as received buffered main

system memory write requests, and then executing the buffered main system memory read request which was determined to be within the predetermined range of address values.

25. A memory control unit as set forth in claim 24 wherein the comparing means includes means for executing an associative comparison between each received system memory read address and all buffered system memory write addresses, if any, to determine if the received system memory read address has an address value within a predetermined range of address values of one of the buffered system memory write addresses, the associative comparison means having an output coupled to said asserting means.

26. A memory control unit as set forth in claim 24 and further comprising:

means coupled to the system memory write request buffering means for determining when a predetermined number of system memory write requests are buffered within the system memory write request buffering means, the predetermined number being less than a maximum capacity of the system memory write request buffering means;

means responsive to the operation of the predetermined number determining means for asserting a signal on the system bus when the predetermined number determining means determines that the predetermined number of system memory write requests are buffered, an assertion of the system bus signal preventing bus connections from issuing further system memory read or system memory write requests on the system bus; and wherein

the executing means is coupled to and is responsive to the operation of the predetermined number determining means for executing at least one buffered system memory write request such that the remaining number of buffered system memory write requests is less than the predetermined number.

* * * * *